

Alena Kovárová

SPECIAL INTERACTION APPROACHES AND THEIR IMPACT ON USABILITY

Dissertation Thesis

FIIT- 10890- 38836

This thesis is submitted in fulfillment of the requirements for the degree of Philosophiæ Doctor (PhD.) in the field of Applied Informatics.

Supervisor: Assoc. Prof. Martin ŠPERKA
Study program: Applied Informatics
Field of study: 9.2.9 Applied Informatics
Institute of Applied Informatics
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
August 2011
Bratislava

Alena Kovárová

ŠPECIÁLNE PRÍSTUPY K INTERAKCII A ICH VPLYV NA POUŽITEĽNOSŤ

Dizertačná práca
FIIT- 10890- 38836

Táto práca je predložená pre naplnenie jednej z podmienok na získanie vedecko-akademickej hodnosti Philosophiæ Doctor v odbore doktorandského štúdia Aplikovaná informatika.

Školiteľ:	doc. Ing. Martin ŠPERKA, PhD.
Forma štúdia:	denná
Začiatok štúdia:	1.10.2004
Študijný program:	Aplikovaná informatika
Študijný odbor:	9.2.9 Aplikovaná informatika
Ústav:	Ústav aplikovanej informatiky
Dátum:	August 2011
Mesto:	Bratislava

Author Alena Kovárová

Supervisor Assoc. Prof. Martin Šperka
 (Slovak University of Technology in Bratislava)
 (now Pan European University, Bratislava)

Reviewers Dr. Božena Mannová
 (Czech Technical University in Prague)
 Assoc. Prof. Andrej Ferko
 (Comenius University in Bratislava)

Keywords Interaction, User Interface, Laser Pointer, Adaptive,
 Personalized, Desktop widget, Departures, Solid
 Geometry, Education, 3D

ACM Subject Classification:

- H.3.4 [**Information Storage and Retrieval**]: Systems and Software —
User profiles and alert services;
- H.5.1 [**Information Systems**] INFORMATION INTERFACES AND
PRESENTATION – Multimedia Information Systems; Artificial,
augmented, and virtual realities
- H.5.2 [**Information Systems**] INFORMATION INTERFACES AND
PRESENTATION – User Interfaces (D.2.2, H.1.2, I.3.6): Graphical
user interfaces (GUI); Input devices and strategies; Interaction
styles
- G.4 [**Mathematics of Computing**] MATHEMATICAL SOFTWARE –
User interfaces

© 2011 Alena Kovárová, Slovak University of Technology in Bratislava

Autorka Alena Kovárová

Školiteľ doc. Ing. Martin ŠPERKA, PhD.
(Slovenská technická univerzita v Bratislave)
(t. č. Paneurópska vysoká škola, Bratislava)

Oponenti Ing. Božena Mannová, Ph.D.
(České vysoké učení technické v Praze)
Doc. RNDr. Andrej Ferko, PhD.
(Univerzita Komenského v Bratislave)

Kľúčové slová Interakcia, používateľské rozhranie, laserové ukazovadlo,
adaptívny, personalizovaný, desktop widget, odchody,
stereometria, vzdelávanie, 3D

ACM Subject Classification:

H.3.4 [**Information Storage and Retrieval**]: Systems and Software —
User profiles and alert services;

H.5.1 [**Information Systems**] INFORMATION INTERFACES AND
PRESENTATION – Multimedia Information Systems; Artificial,
augmented, and virtual realities

H.5.2 [**Information Systems**] INFORMATION INTERFACES AND
PRESENTATION – User Interfaces (D.2.2, H.1.2, I.3.6): Graphical
user interfaces (GUI); Input devices and strategies; Interaction
styles

G.4 [**Mathematics of Computing**] MATHEMATICAL SOFTWARE –
User interfaces

© 2011 Alena Kovárová, Slovenská technická univerzita v Bratislave

Declaration on Word of Honor

I, Alena Kovárová, declare on my honor that I wrote the present thesis using the knowledge obtained during the course of my study. Where information has been derived from other sources, I confirm that this has been indicated in this thesis.

Bratislava 11.8.2011

.....
Author's signature

Annotation

Slovak University of Technology in Bratislava

FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Study program: Applied Informatics

Field of study: Applied Informatics

Author: MSc. Alena Kovárová

Dissertation Thesis: Special Interaction Approaches and their Impact on Usability

Supervisor: Assoc. Prof. Martin ŠPERKA

2011, August

The present work examines various interaction styles. It is divided into three parts, introducing three proposals for enhancement of existing interaction methods.

The first proposal focuses on applying the common hardware to a new style of interaction – the interaction with objects in the outdoor environment. To verify this method we have implemented the pilot system icPoint, which enables the user to interact with a night sky.

The second proposal relates to the streamlining of the information retrieving from the Internet. In this part we discuss the situation, in which the user knows exactly where the required information is located. We propose a method for accelerating the process of obtaining this information based on the user model utilization. In order to verify this method we have implemented a widget, which assists in searching for departure times of public transportation. We have achieved considerably better times in obtaining the requested information.

The third part deals with the enhancement of a graphic user interface for educational 3d graphical editors (in the field of solid geometry), where we focus mainly on a preview hint visualized before an action is executed. This preview shows the consequence of the user's potential action. To verify our theory we have implemented a simulation of the cube cross section, which is a part of secondary school curriculum. Tests performed by users demonstrated that this preview is a useful and desired element of the interface.

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLOGIÍ
Študijný program: Aplikovaná informatika
Študijný odbor: 9.2.9 Aplikovaná informatika

Autor: Mgr. Alena Kovárová
Dizertačná práca: Špeciálne prístupy k interakcii a ich vplyv na
použiteľnosť
Školiteľ práce: doc. Ing. Martin ŠPERKA, PhD.
August 2011

Predkladaná práca sa zaoberá rôznymi druhmi interakcií. Je rozdelená do troch častí, v ktorých prináša tri rôzne návrhy pre vylepšenie doteraz existujúcich metód interakcie.

Prvé vylepšenie sa týka použitia bežne dostupného hardvéru na nový štýl interakcie – interakciu s objektmi mimo uzavretých priestorov. Na overenie tejto metódy sme implementovali pilotný systém icPoint, ktorý používateľovi umožňuje interagovať s hviezdou oblohou.

Druhé vylepšenie sa týka zefektívnenia procesu získavania informácií z Internetu. Tu sa zaoberáme prípadom, kedy používateľ presne vie, kde sa hľadaná informácia nachádza a navrhujeme metódu založenú na využití modelu používateľa, ktorá získanie tejto informácie urýchľuje. Pre overenie sme implementovali widget podporujúci vyhľadávanie odchodov spojov MHD, pri ktorom sme dosiahli mnohonásobné zrýchlenie získavania hľadanej informácie.

Tretia časť sa týka vylepšenie grafického rozhrania pre vzdelávacie 3D grafické editory (oblasť stereometrie), kde sa zameriavame predovšetkým na zobrazovanie náhľadu pred vykonaním akcie. Tento náhľad zobrazuje následok potenciálnej používateľovej akcie. Pre overenie sme implementovali simuláciu rysovania rezu kocky, ktorý je súčasťou stredoškolského učiva. Testovanie používateľmi ukázalo, že tento náhľad je vítanou súčasťou rozhrania.

Acknowledgement

I wish to thank all the people who supported me during my doctoral study and had contributed to finishing this thesis. First of all I would like to thank my supervisor Assoc. Prof. Martin Šperka, who accepted me for the doctoral study and was very forthcoming during the whole seven year period. I would like to thank my colleagues, who helped me to explore the world of informatics, so I could advance in this field. I must thank also the management, which did not lose the belief in me, despite my delays, namely Prof. Mária Bieliková, who always encouraged me. As regards my scientific “coming of age”, I express my thanks to my colleague and good friend Dr. Michal Tvarožek, who never spared me a criticism and offered an abundance of sound advice. My thanks belongs also to Dr. Iveta Kohanová for professional consultations in the field of pedagogy of mathematics.

I would like to thank also all my students, who were inspired by my thoughts and worked under my supervision and thus contributed to this work. I would particularly mention Ing. Lucia Szalayová, Ing. Michal Sokolský and the boys from the Netrollers team: Ing. Michal Dobiš, Ing. Vladimír Hlaváček, Ing. Linh Hoang Xuan, Ing. Michal Jajcaj, Ing. Dušan Lamoš.

My thanks to all my friends who supported me, prayed for me, helped me or encouraged me, as well as to those who took part in the conducted tests. I thank Bob Stump, who was always willing to hear about my research and often made English corrections and Dr. Marián Marton, whose willingness to help out knows no bounds.

I appreciate feedback and comments received from the reviewers, Dr. Božena Mannová and Assoc. Prof. Andrej Ferko, which helped me to improve quality of this work.

Last, but not least, I would like to thank my parents who provided me a background which was essential for me to be able to focus on my research.

About the Author

Alena Kovárová was born in Bratislava, Slovakia on 4 June 1981. She received her Master degree in Mathematics with major in Computer Graphics and minor in Financial Mathematics (2004) from the Comenius University in Bratislava. She is presently a PhD student at the Slovak University of Technology, at Faculty of Informatics and Information Technologies in the field of Applied Informatics. Her research interests are in the area of Human Computer Interaction, with a special focus on usability.



She has published and presented her research results at international as well as local and national conferences. The full list of publications is in Appendix A.1 and awards in A.2.

Her other interests include singing, playing musical instruments, painting and cycling.

List of Tables and Figures

Tables

Table 1: Comparison of usability evaluation methods (Hom, 1996-2003)	14
Table 2: Comparison of usability evaluation methods (Dix, Finlay, Abword, & Beale, 2004)	15
Table 3: Comparison of different widget engines according to the operating system they run and language they can be programmed in.....	46
Table 4: The default controls in Doom for the most often-used functions	49
Table 5: View and scene manipulation differences in five 3D graphical editors .	52
Table 6: Object manipulation differences in five 3D graphical editors.....	53
Table 7: The required number of interactions steps for retrieving time of departure	93
Table 8: Voice commands for basic icPoint screen	i
Table 9: Voice commands for icPoint left side panel, which contains information about a sky object	i

Figures

Figure 1: Two views on acceptance	16
Figure 2: A model for the Unified Theory of Acceptance and Use of Technology (Venkatesh, Morris, Davis, & Davis, 2003).....	18
Figure 3: A model of the attributes of system acceptability (Nielsen, 1993, p. 25)	19
Figure 4: Four categories of pointing at remote object.....	30
Figure 5: The setup of the prototype for Remote pen (left). Student's camera view, where is shown how he draws a diagram representing the direction of the outer product and the magnitude (right) (Ishihara & Ishihara, 2006)	31
Figure 6: Top-projected table hardware configuration (Parker, Mandryk, & Inkpen, 2005).....	32
Figure 7: Vacuum (Bezerianos & Balakrishnan, 2005)	33
Figure 8: Basic object manipulation techniques such as translation (a) and rotation (b) are illustrated in long exposure photographs. Augmentation can be projector-based (a-c) or via video-see-through (d). These application examples show basic augmentations of building structures (a,b,d), distance measurements (c) and material-color simulations (c). (Kurz, Hantsch, Grobe, Schiewe, & Bimber, 2007)	35
Figure 9: The vision based interaction system; The portable LCD projector and USB2.0 camera are placed at the front of the screen, while the user can control the mouse function using a laser pointer (Kim, Lee, Lee, & Lee, 2007)	36
Figure 10: Three types of basic crossing (Shizuki, Hisamatsu, Takahashi, & Tanaka, 2006)	37

Figure 11: Crossing-command mappings for a slideshow and a map viewer application (Shizuki, Hisamatsu, Takahashi, & Tanaka, 2006)	37
Figure 12: An example of applications for outdoor star observation: Google Sky Map (left) and Star Walk (right), both screenshots with inverted colors	38
Figure 13: An illustration of augmented reality without the use of computing resources, only through a glass pane – a terrain with ruins and the glass pane with ruins’ complement (left); observer’s view through the glass pane (right)	39
Figure 14: Spectra sundial (21 st century) (Carmichael, 2011)	40
Figure 15: Examples of desktop widgets screenshots taken from web: Windows Gadgets (a), Google Desktop Gadgets (b), Opera Widgets (c), Yahoo! Widgets (d), and Dashboard (e)	45
Figure 16: Illustration of different editing modes in 3ds Max editor dragging (a), rotating (b) and scaling (c)	54
Figure 17: Screenshot of Archimedes Geo3D application	55
Figure 18: Screenshot of Cabri 3D application	56
Figure 19: Examples of hints for adding a new segment in Cabri 3D	57
Figure 20: A student working with Construct3D in our standard AR lab setup with a head mounted display (Kaufmann, 2009)	58
Figure 21: Screenshots of two tasks with interactive 2D scene from Naucteviac.sk portal (Agemsoft, 2011)	58
Figure 22: A lesson with 3D animation from Naucteviac.sk portal (Agemsoft, 2011)	59
Figure 23: Head mounted laser pointer and detail on laser	62
Figure 24: An example user and devices location within user’s space and environment	63
Figure 25: Outlined camera view	64
Figure 26: Horizontal coordinate system with origin in C	65
Figure 27: Auto-calibration screen with four different colors in rectangular areas serving for detection of screen angle within camera image	70
Figure 28: Outdoor experimental interaction prototyping – a user is lying under a glass pane placed on cardboard boxes aiming a laser beam to a star	72
Figure 29: Screenshot of icPoint project with main screen and left side panel, which contains multimedia information about a selected sky object	74
Figure 30: Screenshot of icPoint project with the main screen displaying a part of the night sky with the selected star in the middle; four large buttons placed at the bottom allow controlling icPoint by a laser pointer; panel on the right side contains application settings	76
Figure 31: Hardware usage for icPoint – comfortable usage of table with transparent glass	77
Figure 32: Outdoor interaction with remote objects – a user is lying under a glass table and aiming a laser beam at a star	77
Figure 33: Overview of our personalized interaction approach	82
Figure 34: Ontology model of data from public transportation departures	85
Figure 35: Flowchart for estimation of user’s choices	87
Figure 36: Widget layout description	90

Figure 37: Widget setup for multiple lines within one route (in Slovak language, translation of route: Home -> Work)	91
Figure 38: Conceptual architecture of the public transportation departures widget	91
Figure 39: Time consumption comparison for obtaining information from various sources using different ways to speed up search.....	94
Figure 40: Screenshot of the iTransit application for iPhones: Screen with the closest departures from the closest stops according to actual time and the user's GPS position	97
Figure 41: Cube with 3 points	101
Figure 42: Screenshot of our pilot application Stereo3D.....	105
Figure 43: Screenshot of our pilot application InteractiveCube	105
Figure 44: Context menu for a line on Stereo3D	106
Figure 45: Preview of segment extension in InteractiveCube	106
Figure 46: Graph showing dependency between number of test users and found usability problems (Nielsen & Landauer, 1993).....	g
Figure 47: Physical data model of the widget database	k
Figure 48: An example of the first construction rule	r
Figure 49: An example of the second construction rule.....	s
Figure 50: An example of the third construction rule	t

List of abbreviations

2D: two-dimensional	26, 32, 50, 53, 54, 58, 68, 101
3D: three-dimensional.....	22, 25, 26, 27, 32, 33, 34, 48, 49, 50, 51, 54, 58, 63, 74, 101, 103
6DoF: six degrees of freedom	49
API: Application Programming Interface	44, 45
AR: Augmented Reality	35, 57
AU: Average users	108, 109, 110, 111
CCD: Charge Coupled Device - a major technology for digital imaging.....	31, 34
FPS: First Person Shooter	48, 49, 50
FT: Future teachers.....	108, 109, 110, 111, 1
GPS: Global Positioning System	27, 38, 68, 77, 83
GUI: Graphical User Interface.....	22, 24, 25, 70, 91, 92, 98
HCI: Human Computer Interaction	9, 21
HS: High school students	108, 109, 111, 1
IO: Input/Output	21, 23, 26, 27
KLM: Keystroke-Level Model	13, 88, 92
LED: Light-Emitting Diode.....	34
TRA: Theory of Reasoned Action	17
UI: User Interface	7, 8, 9, 11, 22, 23, 26, 107, 115
UTAUT: Theory of Acceptance and Use of Technology	17, 18
WASD: key combination - W moves forward, S moves backward, A strafes left, and D strafes right	50, 51, 59
WIMP: windows, icons, menus, and a pointing device.....	22, 24, 58

Contents

1.	Introduction	2
2.	Usability and Acceptance in Human-computer Interaction	6
2.1	Usability	6
2.1.1	Usability Inspection Methods	9
2.1.2	Usability Testing Methods	11
2.1.3	Inquiry Methods	12
2.1.4	Analytical Modeling Methods	13
2.1.5	Summary and Comparison of Evaluation Methods	13
2.2	Acceptance	15
2.2.1	User Acceptance Testing	16
2.2.2	Theories on Acceptance and Use of New Technologies	16
2.2.3	User Interface Design	19
3.	Interaction Approaches	21
3.1	Interaction Styles	22
3.2	Rules for a Good User Interface	23
3.3	Hardware Supported Interaction	24
3.3.1	Other Input and Output Devices	27
4.	Current Interaction Approaches	29
4.1	Remote Object Interaction	29
4.1.1	Interaction from Indoor to Remote Indoor	30
4.1.2	Indoor Interaction within Table Distances	31
4.1.3	Laser Pointer	33
4.1.4	Outdoor Interaction	38
4.1.5	Summary	40
4.2	Lowering the Demands on the User's Interaction	41
4.2.1	Fast Information Retrieval Through Web Browsers	42
4.2.2	Widgets and Gadgets	43
4.2.3	Summary	46
4.3	Interaction in 3D Environments Using a Typical PC	48
4.3.1	3D games	48
4.3.2	Commercial and Free 3D Graphic Editors	51
4.3.3	Educational 3D Editors	54
4.3.4	Summary	59
5.	Interaction with Distant Objects	61

5.1	The User's Line of Sight Calculation	61
5.2	Interaction with Computer without Mouse and Keyboard	68
5.2.1	Laser Pointer as a Mouse	69
5.2.2	Voice Commands	71
5.3	Object of Interest Identification	71
5.3.1	Application Domain	71
5.3.2	Finding Sky Objects	72
5.3.3	Information about Sky Objects	74
5.4	Evaluation	75
5.4.1	System Description	75
5.4.2	Experiment	78
5.5	Discussion	79
6.	Interaction Support Based on User Model	81
6.1	Reducing the Number of the User's Interaction Steps	81
6.2	Personalized Information Retrieving for Public Transportation Departures	83
6.2.1	Application Domain	83
6.2.2	Personalization	86
6.3	Evaluation	88
6.3.1	System Description	88
6.3.2	Experiment	92
6.4	Discussion	96
7.	Interaction Using Visual Help in Simple 3D Editors	98
7.1	Improvements of GUI Elements	98
7.1.1	Application Domain	101
7.2	Evaluation	103
7.2.1	Systems Descriptions	103
7.2.2	Experiments	106
7.2.3	Test results	109
7.3	Discussion	112
8.	Conclusions	114
	References	118
	Appendix A	a
A.1	Publications	a
A.2	Awards	c
A.3	Research Projects	c
A.4	Supervised Theses and Projects	d
	Appendix B	g

B.1	Number of Test Users	g
B.2	icPoint	h
B.3	Widget	j
Appendix C		l
C.1	Feedback questionnaires	l
C.2	Graph	p
C.3	Cube Section Construction Method	q

1. Introduction

There are many different kinds of interaction with computers - from devices such as a mouse and a keyboard, through different types of menus and toolbars, up to mouse gesture recognition or eye movement capturing. The dialog between a person and a computer has a history as long as computers themselves. This communication depends on the technical progress in each time period. Today a user can buy a variety of devices that can handle various specific situations. Some of such devices even contain an embedded system and thus often replace personal computers (e.g. cell phones). Even though such devices offer a very interesting area of research, our research has focused on conventional personal computers (desktops or portables), the most often used input and output devices¹ and commonly available (and affordable) tools. Since the potential of these devices is not yet fully explored, our focus was on the possibilities of interaction improvement. During our research we experimented with different types of input and output techniques and this thesis presents those that we consider the most interesting.

Our very first idea was to invent and to implement a better interaction in any possible way. Our focus was mainly on the educational domain where we tried to create a more intuitive (natural) interaction. We identified three key areas within the human-computer interaction field, where we applied our effort:

1. Cover new domains of interaction by using standard equipment in new ways.
2. Explore a specific type of domain which has the potential of reducing the number of interaction steps.
3. Apply already known principles from one domain to another.

¹ such as a mouse, a keyboard, loudspeakers or headphones, a microphone or a headset and a web-camera

We tried to determine which application domain is optimal for each of the specified areas. We preferred educational areas. Finally, we chose the following domains and their corresponding goals according to SIGCHI categorization (Hewett, et al., 2009).

1. There are many different domains where people do not use a computer because it is difficult to transfer the user's tasks to the computer. One of these domains is the situation where the user wants to point at a remote object off the computer screen as input information for the computer. An example of this situation is a man standing on an elevated site above the town, taking his/her computer, pointing to a building and wanting to know/hear its name or possibly any other relevant information about it. A similar situation is in the mountains, where a person wants to know the name of a peak he/she is pointing to. Another example is a man lying down on his/her back in the night watching stars, wanting to know the name of the one to which he/she is pointing. (Here we did not consider a solution requiring an accelerometer, as it is neither a common nor affordable part for computers.)

Goal 1: To propose and verify a new method of computer aided interaction with remote objects within an outdoor environment using common and affordable equipment (including input and output devices).

This goal belongs to the following SIGCHI categories: Dialogue Techniques: Dialogue Inputs: Input techniques – pointing and selection; Dialogue Interaction Techniques: Dialogue type and techniques.

2. Due to the continually growing volume of information that is made freely available online, people often find themselves in the inconvenient situation where they have to invest a disproportional effort and time to gain the information they need. This process includes for example decisions such as which electronic newspaper to read, which sports section to monitor, which broadcast to watch, which web pages contain relevant information or simply whether the needed information is worth the time and effort. This is of course a daily struggle; most of us would appreciate the time-saving and effort-saving option of having this “personalized” information wait for us somewhere nicely aligned. To be as close as possible to this vision, people came to the point of choosing a favorite newspaper, favorite channels and programs, favorite web pages; simply said: favorite information sources. But this is still not enough;

even within these favorite sources it is still necessary to search and to filter. This simply reflects the fact that the majority of the available information sources are built for the masses and therefore do not have any implemented personalization / personal adaptation features to serve the needs of each individual person.

People do a considerable part of such information retrieval conditionally or repetitively or even regularly. This allows us to capture, extract (discover patterns), store and finally use this user's behavior to estimate / predict the user's needs. Applying these predictions, the user can obtain the needed information with a fewer number of actions or even without a single click.

Goal 2: To propose and verify a method, which on the basis of observing the users' actions, stores his/her choices and thus reduces the demands on the user's interaction when retrieving web information.

Category: Human-Machine Fit and Adaptation: System adaptation: customization and tailorability techniques.

3. A lot of applications with wide functionality deal with the problem of how to offer this functionality. How does the application let users know what they can do with an already selected tool or mode? The often observed behavior is that the users try to use it. By trial and error they discover its functionality, especially if they are not motivated to read a manual or even tooltips and they do not have experience with anything similar. Although it can be a very useful way of learning to control the application, we consider it slow. We suggested visualizing each type of information in different, more or less known forms. A new idea on what to visualize and how, can be found for example in Microsoft Word 2007, where after hovering over a format, a preview of reformatted text appears and on rollout it restores to the original formatting. This means that the consequence of a selection is visible before the selection itself. This idea can be used also in other domains. We chose the domain of graphic editors, where to create a visible consequence is not as trivial as with formatting the text. Such a solution has potential to be used in educational applications dealing with (technical) drawing.

Goal 3: To verify whether different methods of visualized information increase usability of 3D graphical editors, with emphasis on graphical hint for a hovered object within a graphic editor, where this suggestion visualizes the consequence of object's selection.

This goal belongs to the following SIGCHI category: Dialogue Techniques: Dialogue Outputs: Screen layout issues

The work is organized as follows. In chapter 2 we provide an introduction to the topic of usability and in chapter 3 to interaction approaches. These two chapters contain brief definitions or descriptions of concepts used in our research. Chapter 4 is devoted to related work connected to all three goals, after which follow chapters 5, 6 and 7, each focusing on one of our goals, their description and evaluation. The work is closed in chapter 8, where we summarize our contributions and outline future work.

2. Usability and Acceptance in Human-computer Interaction

When working with human-computer interaction, whether it is proposing an unusual way of control, eliminating the number of interaction steps or a more suggestive user interface, all these approaches deal with the question: “Will users like this?” What exactly the word “like” means and how it depends on other factors is a matter of usability and acceptance. In the next sections, three basic characteristics of usability are cited, followed by a list of usability evaluation methods. Only the most often used or the ones we have used in our experiments are briefly explained. These and other methods are presented in the form of two comparative tables, from which their differences can be seen. A later subsection presents Nielsen’s explanation of “Why you only need to test with five users.” The chapter ends with a brief history of acceptance theories, which summarize all “personal and social” influence factors. These have to be considered in usability evaluations.

2.1 Usability

In human-computer interaction or computer science, usability is often explained as the user-friendliness or ease of use of a given computer system. The ISO 9241-11 standard defines usability as “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.” There are several points of view which describe the conditions under which these qualities are achieved:

Consultant Jakob Nielsen and computer science professor Ben Shneiderman state that the usability of a system predicts five factors of good system design (Nielsen, 1993, p. 26) (Shneiderman, 1980):

1. **Learnability:** How easy is it for users to learn the system, and accomplish basic tasks the first time they encounter the system?
2. **Efficiency:** Once users have learned to operate the system, how quickly can they perform their tasks? If the system is designed to replace a legacy system, how does the new system affect task efficiency?
3. **Memorability:** After a period of non-use, how easy is it for users to return to the system and resume their tasks?
4. **Errors:** How likely are errors to occur within the system, user-generated or system-generated? How severe are these errors, and how easily can users recover from the errors?
5. **Satisfaction:** How pleasant is it to use the system? The satisfaction of a user is often directly correlated with other concepts of usability (learnability, efficiency, memorability, and error handling).

To evaluate the usability of a system, both the user interface (UI) and functionality must be considered. An intuitive UI can lower the learning curve of the system, and increase the efficiency of typical tasks, but it must offer sufficient functionality to remain useful. User interface engineering has been subject to extensive research, and many guidelines and principles have been proposed to improve the quality of UI design. Raskin in his book *The Humane Interface* (Raskin, 2000) suggests two paramount laws of UI design:

- **First Law:** A computer shall not harm your work or, through inactivity, allow your work to come to harm.
- **Second Law:** A computer shall not waste your time or require you to do more work than is strictly necessary.

In other words, users should be able to perform their tasks in an efficient manner without any interruption. The computer system should work with them – not against them.

Larry Constantine and Lucy Lockwood in their usage-centered design suggest that UI design should be directed by 6 principles (Constantine & Lockwood, 1999) (Constantine & Lockwood, 1996):

1. **The structure principle:** Design should organize the UI purposefully, in meaningful and useful ways based on clear, consistent models that are apparent and recognizable to users,

putting related things together and separating unrelated things, differentiating dissimilar things and making similar things resemble one another. The structure principle is concerned with overall UI architecture.

2. **The simplicity principle:** The design should make simple, common tasks easy, communicating clearly and simply in the user's own language, and providing good shortcuts that are meaningfully related to longer procedures.
3. **The visibility principle:** The design should make all needed options and materials for a given task visible without distracting the user with extraneous or redundant information. Good designs do not overwhelm users with alternatives or confuse with unneeded information.
4. **The feedback principle:** The design should keep users informed of actions or interpretations, changes of state or condition, and errors or exceptions that are relevant and of interest to the user through clear, concise, and unambiguous language familiar to users.
5. **The tolerance principle:** The design should be flexible and tolerant, reducing the cost of mistakes and misuse by allowing undoing and redoing, while also preventing errors wherever possible by tolerating varied inputs and sequences and by interpreting all reasonable actions.
6. **The reuse principle:** The design should reuse internal and external components and behaviors, maintaining consistency with purpose rather than merely arbitrary consistency, thus reducing the need for users to rethink and remember.

Nielsen's and Shneiderman's five factors arise directly from the usability definition and give us the more theoretical view. In contrast, Constantine and Lockwood took the definition and searched for causes, which resulted in their six principles and therefore these principles are more practical. These are not the only principles; very similar are Shneiderman's Eight Golden Rules of interface design (Shneiderman, 1998). And it is natural that the very same pattern can be found even in evaluation methods, for example in Nielsen's 10 usability heuristics (see page 10).

These general factors, laws and principles work as guidelines and there also has to be methods which can evaluate or measure system usability. There are four basic ways of evaluating UIs (Nielsen & Mack, 1994):

1. **Automatically** – usability measures computed by running a UI specification through some program
2. **Empirically** – usability assessed by testing the interface with real users
3. **Formally** – using exact models and formulas to calculate usability measures
4. **Informally** – based on rules of thumb and the general skill and experience of the evaluators

Under the current state of the art, automatic methods² are limited and formal methods are very difficult to apply and do not scale up well to handle larger UIs. Empirical methods are the main way of evaluating UIs, with user testing probably being the most commonly used method. Often, real users can be difficult or expensive to recruit in sufficient numbers to test all aspects of all the versions of an evolving design, leading to the use of inspection as a way to "save users." Several studies have shown that usability inspection methods are able to find many usability problems that are overlooked by user testing but that user testing also finds some problems that are overlooked by inspection, meaning that the best results can often be achieved by combining several methods.

We were using these types of evaluation during UI development, therefore in the next subsections a more detailed description follows, mostly compiled from the HCI book (Dix, Finlay, Abword, & Beale, 2004) and Usability Engineering book (Nielsen, 1993) and James Hom's usability web site (Hom, 1996-2003).

2.1.1 Usability Inspection Methods

Usability inspection³ is a set of informal methods that are all based on having evaluators inspect the interface. It is often used in cases where there is no alternative such as user testing. Typically, usability inspection is aimed at finding usability problems in a design, its severity and the overall usability of an entire design. Many inspection methods can be performed early in the usability engineering lifecycle.

In 1993 Dix et al. (Dix, Finlay, Abword, & Beale, 2004) categorized these methods in following 4 classes: *Heuristic Evaluation*, *Cognitive walkthrough*, *Model-Based Evaluation* and *Evaluations using previous studies*. Later, in 1994, Nielsen and Mack published

² E.g. Automated expert review

³ Also called Usability Evaluation, Expert-based Interface Evaluation, Analytical Methods, Expert Review or Evaluation through Expert Analysis

categorization dividing these methods in 7 classes where the first two are the same and the next five are: *Formal Usability Inspections*, *Pluralistic Walkthroughs*, *Feature Inspection*, *Consistency Inspection*, and *Standards Inspection* (Nielsen & Mack, Usability Inspection Method, 1994). In 2001 Ivory and Hearst enriched this list by *Perspective-Based Inspection* and *Guideline Review* and created a new class from Model-Based Evaluation named *Analytical Modeling Methods* (see section 2.1.4) (Ivory & Hearst, 2001). Later on, also these methods were developed: *Guideline Checklist*, *Card Sorts*, *Tree Tests*, *Activity Analysis*, etc. The most known are shortly described in following lines with accent on those, we used in our research. Comparison can be found in Table 1 and Table 2.

Heuristic Evaluation

Heuristic evaluation⁴ involves setting up a team of 3-5 independent evaluators, who systematically inspect the design by means of broad guidelines for good practice. A well-known set of heuristics is the one proposed by Nielsen⁵ (Nielsen & Molich, 1990):

- Visibility of system status - provide feedback
- Match between system and the real world - speak the user's language
- User control and freedom - provide clearly-marked exits, including undo and redo
- Consistency and standards
- Error prevention
- Recognition rather than recall - minimize user memory load by making objects, actions and options visible
- Flexibility and efficiency of use - provide short cuts
- Aesthetic and minimalist design - simple and natural dialogue
- Help users recognize, diagnose, and recover from errors
- Help and documentation

Cognitive Walkthrough

In cognitive walkthrough (Lewis, Polson, Wharton, & Rieman, 1990) an analyst goes through all of the possible courses of action the user might take and at each of them analyzes these three points:

⁴ Often considered as one of Discount Usability Technique

⁵ Can be found also at Nielsen's web page

http://www.useit.com/papers/heuristic/heuristic_list.html

- Will the correct action be made sufficiently evident to users?
- Will users connect the correct action's description with what they are trying to achieve?
- Will users interpret the system's response to the chosen action correctly?

Any negative answer to these questions means an identified problem.

2.1.2 Usability Testing Methods

Unlike previous methods, the usability testing⁶ is an empirical method and therefore requires observing real users. This is a black-box technique. The goal of these tests is to determine problems which hinder the user from working with greater speed, recall, accuracy, and emotional response⁷.

- **Performance** - How much time, and how many steps, are required for the user to complete basic tasks?
- **Accuracy** - How many mistakes did users make? (fatal or recoverable)
- **Recall** - How much does the user remember afterwards or after periods of non-use?
- **Emotional response** - How does the user feel about the tasks completed (confident, stressed)?

There are several methods which can be used to find out if the UI has these four qualities. Some of them are *Coaching Method*, *Co-discovery Learning (Subjects-in-Tandem)*, *Log File Analysis*, *Performance Measurement*, *Question-asking Protocol (Cooperative Evaluation)*, *Remote Testing*, *Retrospective Testing*, *Shadowing Method*, *Teaching Method*, *Think-aloud Protocol (Protocol Analysis)* (Ivory & Hearst, 2001). Some of other newer methods are: *Component-based Usability Testing*, *Hallway Testing (Hall Intercept Testing)*, *Rapid Iterative Testing and Evaluation (RITE)*. The ones closest to our research or those, we used, are shortly described in the following lines. Comparison can be found in Table 1 and Table 2.

Using the **Hallway testing** method, 5 random people “who pass by in the hallway” are brought in to test the interface. These 5 random users have to be indicative of a cross-section of end users.

⁶ Also known as Empirical Interface Evaluation, Usability Tests, User Evaluation, Observational Techniques or Observational Evaluation Techniques; these methods are derived from experiment and observation rather than theory

⁷ Also known as comfort or satisfaction

Remote testing⁸ involves the use of a specially modified online survey, allowing the quantification of user testing studies by providing the ability to generate large sample sizes. Similar to an in-lab study, a remote usability test is task-based and the platforms allow you to capture clicks and task times. The tests are carried out in the user's own environment (rather than labs) helping further simulate real-life scenario testing. Additionally, this style of user testing also provides an opportunity to segment feedback by demographic, attitudinal and behavioral type.

Performance Measurement is a rigorous usability evaluation of a working system under realistic conditions to identify usability problems. The user is given tasks to complete, and the evaluator measures and compare relevant parameters such as percentage of tasks or subtasks successfully completed (success rate), time required for each task or subtask (task time), frequency and type of errors, and duration of pauses, indications of user frustration, user satisfaction with requirements and the ways in which the user seeks assistance.

2.1.3 Inquiry Methods

Here, usability evaluators obtain information about users' likes, dislikes, needs, and understanding of the system by talking to them, observing them using the system in real work (not for the purpose of usability testing), or letting them answer questions verbally or in written form. Inquiry methods include: *Contextual Inquiry*, *Field Observation/Ethnographic Study*, *Focus Groups*, *Interviews*, *Logging Actual Use*, *Proactive Field Study*, *Questionnaires*, *Screen Snapshots*, *Self-Reporting Logs*, *Surveys*, *Task/Action Analysis*, *User Feedback*, etc. (Ivory & Hearst, 2001) (Hom, 1996-2003). Comparison of the most used methods can be found in Table 1 and Table 2.

Logging involves having the computer automatically collect statistics about the detailed use of the system. Typically, an interface log will contain statistics about the frequency with which each user has used each feature in the program and the frequency with which various events of interest have occurred. Statistics showing the frequency of use of commands and other system features can be used to optimize frequently used features and to identify the features that are rarely used or not used. Statistics showing the frequency of various error situations and the use of online help can be used to improve the usability of future

⁸ Also known as Unmoderated or Asynchronous Usability Testing

releases of the system by redesigning the features causing the most errors and most access for online help. This technique can be used at the test or deployment stages of software development.

2.1.4 Analytical Modeling Methods

According Dix et al. (Dix, Finlay, Abword, & Beale, 2004) the Model-Based Evaluation covers different types of models, which model some aspect of user's understanding, knowledge, intention or processing. They categorized cognitive models as follows:

- Hierarchical representation of the user's task and goal structure – here belongs, e.g., GOMS (*Goals, Operators, Methods and Selection*) model and CCT (*Cognitive Complexity Theory*)
- Linguistic and grammatical models – here belongs, e.g., BNF (*Backus-Naur Form*) and TAG (*Task-Action Grammar*)
- Physical and device-level models – here belongs, e.g., KLM (*Keystroke-Level Model*), which gives detailed predictions about user performance – acquisition and execution of simple command sequences taking less than 20 seconds – taking into account the human motor system.

Ivory and Hearst surveyed following Analytical Modeling Methods: *GOMS Analysis*, *UIDE Analysis*, *Cognitive Task Analysis*, *Task-Environment Analysis*, *Knowledge Analysis*, *Design Analysis*, *Programmable User Models* (Ivory & Hearst, 2001).

2.1.5 Summary and Comparison of Evaluation Methods

Every method has its advantages and disadvantages. Some are good for web pages and others are better for office applications. Each method is appropriate in other contexts. We have chosen the method of evaluation according to our needs, which resulted from the type of application that we have implemented as well as the possibilities we have had available for our evaluation (more details can be found in chapters dedicated to our experiments). The two following tables contain a brief comparison of the most often used methods. Table 1 compares methods on the basis of the applicable stages, number and type of needed personnel, which usability issues are covered, if it can be conducted remotely, and if a method can provide quantitative data.

Table 1: Comparison of usability evaluation methods (Hom, 1996-2003)

		Applicable stages					Personnel needed			Usability issues covered			Can be conducted remotely	Can obtain quantitative data
		Requirement	Design	Code	Test	Deployment	Usability experts	Software developers	Users	Effectiveness	Efficiency	Satisfaction		
Inspection	Cognitive Walkthrough	X	✓	✓	✓	✓	1-4	0-2	0	✓	X	X	X	X
	Feature Inspection	X	X	✓	✓	✓	1	0	0	✓	X	X	✓	X
	Heuristic Evaluation	X	✓	✓	✓	✓	4	0	0	✓	✓	X	✓	X
	Pluralistic Walkthrough	X	✓	X	X	X	1	1	2	✓	X	✓	X	X
Testing	Coaching Method	X	✓	✓	✓	✓	1	0	4	✓	X	✓	X	X
	Co-discovery Learning	X	✓	✓	✓	✓	1	0	6	✓	X	✓	X	X
	Performance Measur.	X	✓	✓	✓	✓	1	0	6	✓	✓	X	X	✓
	Question-asking Protocol	X	✓	✓	✓	✓	1	0	4	✓	X	✓	X	X
	Remote Testing	X	✓	✓	✓	✓	1	0	5	✓	✓	✓	✓	✓
	Retrospective Testing	X	✓	✓	✓	✓	1	0	4	✓	✓	✓	X	✓
	Shadowing Method	X	✓	✓	✓	✓	1	0	4	✓	✓	X	X	✓
	Teaching Method	X	✓	✓	✓	✓	1	0	4	✓	X	✓	X	X
	Thinking-aloud Protocol	X	✓	✓	✓	✓	1	0	4	✓	X	✓	X	X
	Field Observation	X	X	X	✓	✓	1	0	2	✓	X	✓	X	X
Inquiry	Focus Groups	X	X	X	✓	✓	1	0	6	✓	X	✓	X	X
	Interviews	X	✓	✓	✓	✓	1	0	2	✓	X	✓	X	X
	Logging Actual Use	X	X	X	✓	✓	1	0	6	✓	✓	X	✓	X
	Proactive Field Study	✓	✓	X	X	X	1	0	2	X	X	X	X	X

The number of test users does not need to be high. Why it is so, is explained in Nielsen's widely cited web-article (Nielsen, 2000): "Why you only need to test with five users." He summarized the past decade's research and brought a mathematical model (find more details in Appendix B, section B.1 Number of Test Users).

Table 2 compares methods based on ten factors that distinguish different evaluation techniques and therefore, help to make an appropriate choice. The factors are: the *stage* in the cycle – from design to full implementation; the *style of evaluation* – laboratory as an unnatural environment with better possibilities to control an experiment or the opposite: field studies; the *objectivity* – how heavily it relies on the interpretation of the evaluator; the *type of measures* provided –

qualitative⁹, quantitative¹⁰, or both; the level of *information* provided – high or low; the *immediacy* of response (recall may be incomplete); the *intrusiveness* – whether users notice tracking, which can influence their behavior; the *time*; the resources – how much *equipment* / time / money / participants / experts are needed; the level of expertise – high, medium, low.

Table 2: Comparison of usability evaluation methods
(Dix, Finlay, Abword, & Beale, 2004)

	Stages	Laboratory Style	Field Style	Objective?	Qualitative Measure	Quantitative Measure	Information level	Immediacy	Intrusive?	Time	Equipment	Expertise
Cognitive walkthrough	All	✓	×	×	✓	×	L	N/A	×	M	L	H
Heuristic evaluation	All	✓	×	×	✓	×	H	N/A	×	L	L	M
Model based	Design	✓	×	×	✓	×	L	N/A	×	M	L	H
Experiment	All	✓	×	✓	×	✓	L/H	✓	✓	H	M	M
Interviews	All	✓	✓	×	✓	✓	H	×	×	L	L	L
Questionnaire	All	✓	✓	×	✓	✓	H	×	×	L	L	L
Think aloud ¹	Impl.	✓	✓	×	✓	×	H/L	✓	✓	H	L	M
Protocol analysis ²	Impl.	✓	✓	×	✓	×	H/L	✓	✓	H	H	H
Post-task walkthrough	Impl.	✓	✓	×	✓	×	H/L	×	×	M	L	M

H - High, M - Medium, L - Low

¹ Assuming a simple paper and pencil record

² Including video, audio and system recording, what is intrusive except system logs

2.2 Acceptance

Dealing with acceptance, there are two views from different sides (see Figure 1). The first is a black-box software testing – the test determine if the requirements on the specification of application are met. The second is derived from social psychology, where different theories discuss what motivates or hinder users to use a new technology.

⁹ Quantitative data help to determine accuracy, speed, and recall. E.g. number of mistakes the user made, time at which the task was performed, time that something was remembered.

¹⁰ Qualitative data - testers describe something, which can be either recorded, or can be gathered through questionnaires with both multiple-choice and open-ended questions.

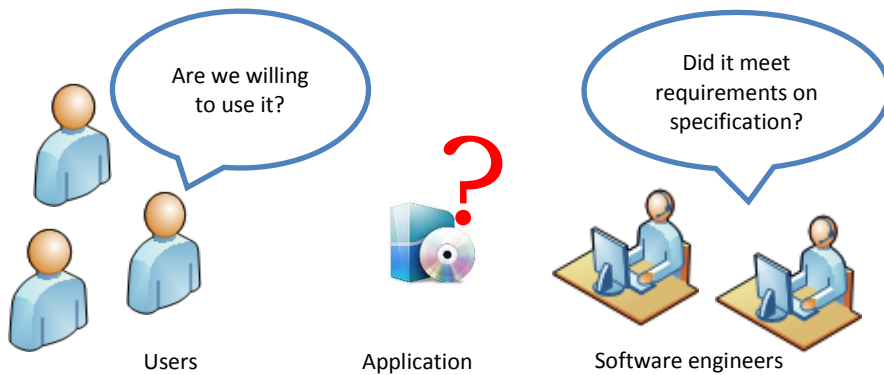


Figure 1: Two views on acceptance

2.2.1 User Acceptance Testing

Acceptance tests are black box system tests. It is a process to obtain confirmation that a system meets mutually agreed-upon requirements. Each acceptance test represents some expected result from the system, e.g., according to the each defined use case. Customers are responsible for verifying the correctness of the acceptance tests. It is usually one of the final stages of a project and often occurs before a client or customer accepts the new system.

This testing process is similar to *functional testing*. The difference is the functional testing can be handled by self-executing scripts or test programs in simulated environment, but acceptance testing must be carried out directly by users in their environment. In both cases, the tests consist of the following five steps:

- The identification of functions that the software is expected to perform
- The creation of input data based on the function's specifications
- The determination of output based on the function's specifications
- The execution of the test case
- The comparison of actual and expected outputs

2.2.2 Theories on Acceptance and Use of New Technologies

It is usually a problem to enforce a new kind of anything, including a new kind of control. People are used to an old style and the older they are, the less they are willing to learn new things. Age isn't the only factor. There are several models and theories, mainly derived from the social psychology field, which summarize these factors.

Theory of Reasoned Action (TRA) (Fishbein & Ajzen, 1975), (Ajzen & Fishbein, Understanding Attitudes and Predicting Social Behavior, 1980) posits that a person's volitional behavior is a function of an individual's attitude towards the behavior and subjective norms surrounding the performance of the behavior. Simply said, if a user intends to do something, then it is likely that he/she will do it. Five years later Ajzen improved his theory and named it the **Theory of Planned Behavior** (Ajzen, 1985), where he suggests, that what a person will do is dependent also on his/her behavioral control, defined as one's perception of the difficulty of performing a behavior, including required effort and resources. Another extension of TRA is the **Technology Acceptance Model** created by Davis (Davis, 1989), (Davis, Bagozzi, & Warshaw, 1989), where he suggests that a person's intention to use a system is determined by perceived usefulness and perceived ease of use.

Innovation Diffusion Theory proposed by Rogers (Rogers, 1995) suggested that the rate of adoption of innovations is impacted by five factors: relative advantage, compatibility, trialability, observability, and complexity. This list was later expanded in the context of IS research to include voluntariness, image, ease of use, result demonstrability, and visibility (Moore & Benbasat, 1991).

All these theories and several others were unified by Venkatesh et al. (Venkatesh, Morris, Davis, & Davis, 2003) in **Unified Theory of Acceptance and Use of Technology** (UTAUT). Venkatesh et al. extend Davis's model to take into account four new constructs (performance expectancy, effort expectancy, social influence and facilitating conditions) that bear significant influence on behavioral intention and ultimately usage of technologies. The variables of gender, age, experience and voluntariness of use are posited to mediate the impact of the four key constructs on usage intention and behavior as shown on Figure 2.

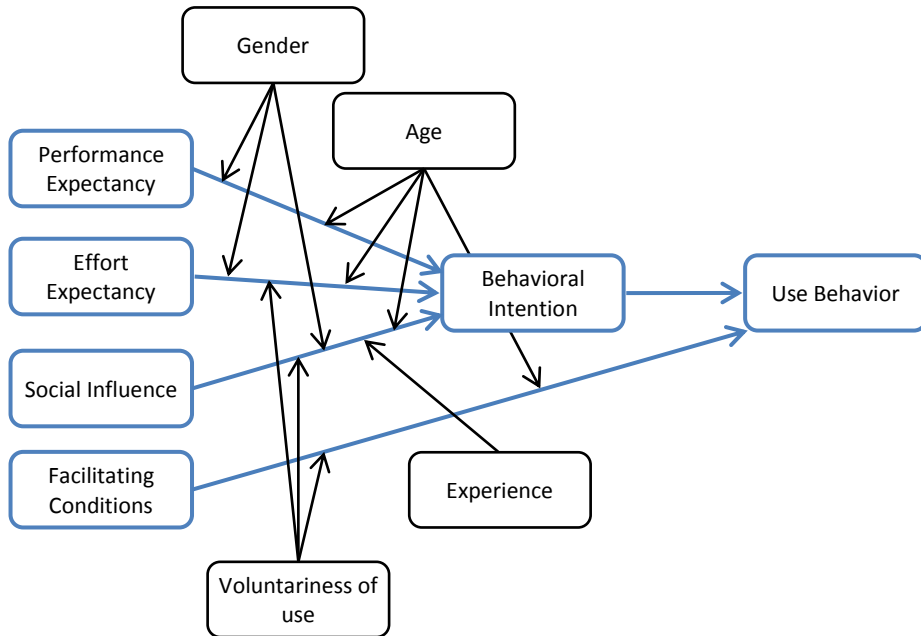


Figure 2: A model for the Unified Theory of Acceptance and Use of Technology (Venkatesh, Morris, Davis, & Davis, 2003)

Venkatesh defined his four constructs as follows:

- **Performance expectancy** - The degree to which an individual believes that using the system will help him/her to attain gains in job performance.
- **Effort expectancy** - The degree of ease associated with the use of the system.
- **Social influence** - The degree to which an individual perceives that important others believe he/she should use the new system.
- **Facilitating conditions** - The degree to which an individual believes that an organizational and technical infrastructure exists to support use of the system.

UTAUT very precisely describes user's intentions and influences which form his/her final willingness to use a system. These influences can be different for each user, thus we included in our questionnaires questions revealing them.

However, there are designers of systems who understand the problem of system acceptability from another point of view. Using this

view, the system acceptability attributes are determined and divided as depicted in Figure 3:

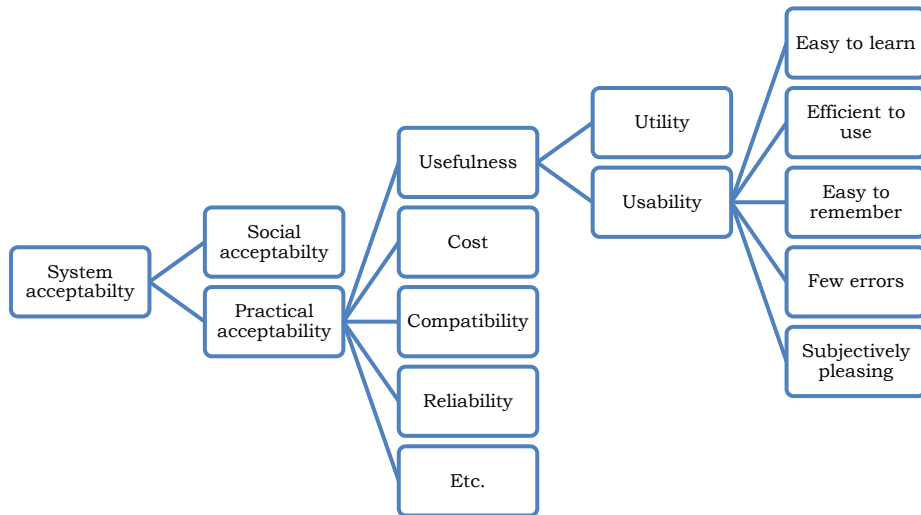


Figure 3: A model of the attributes of system acceptability
(Nielsen, 1993, p. 25)

Here the far right side of the model consists of the five factors of a good system design (described on page 7). In our research we focused mainly on the usability, but we also took into account the rest of the attributes of practical acceptability – affordable and common equipment lower the costs and increase compatibility (in our case, reliability is not a present attribute). An exactly specified group of users and our focus on their needs results in a high degree of utility.

2.2.3 User Interface Design

Since there are many different ways to provide needed functionality, designers can never be sure whether they chose the right one. To minimize the risk of unfulfilled users' expectations, there are five fundamental features of design usability (Preece & Keller, 1990, p. 34):

1. **User centered design** – focused from the start on the user and tasks
2. **Participative design** – with the users as members of the design team
3. **Experimental design** – with formal user tests of usability in pilot trials simulations and full prototype evaluations
4. **Iterative design** – design, test and measure, and redesign as a regular cycle until results satisfy usability specification

5. **User-supportive design** – training, selection (when appropriate) manuals, quick reference card, aid to 'local experts, 'help' systems, e.g. on-line (context-specific help) of off-line ('hot-line' phone service)

During our development process, we always focused from the start on users and their tasks. We, developers, always fit into the target group of users and thus, we have users also in our design team. While prototyping, we collected reservations about the functionality and its method of delivery. These were included in the iterative design process. Finally we created manuals.

3. Interaction Approaches

To find a better interaction approach, a summary is needed of all those interaction styles, types, methods and devices that are already used, whether commonly or rarely. In the first section we started with the list of interaction styles. Since the interaction itself is limited by the input and output devices (IO devices) that can be used, in the second section we mention different types of devices with emphasis on the usual ones (keyboard, mouse, monitor, web-cam, microphone, loudspeakers), where we summarize their interaction possibilities and their intuitiveness. The next section is focused on the interaction possibilities provided by software – the GUIs.

The term ***intuitive*** (or natural) ***interaction*** does not have an exact meaning in connection with HCI. We defined it for the purpose of this thesis as an interaction that is parallel to interaction in the real world whether it is human-human interaction or human-object (environment) interaction. For example, this includes speaking, pointing, typing and face gestures.

The term ***direct interaction***¹¹ also does not have an exact meaning and we defined it as an interaction where a user's input (as an action) has the same localization as a system's output (as a reaction) within a short time (it is similar to "direct manipulation," but the localization is added). E.g. if the user touches something, the direct interaction would be any change in the place of the touch, visual or other.

¹¹ Sometimes known as "natural dialog" concept or "direct control"

3.1 Interaction Styles

According to Dix et al. (Dix, Finlay, Abword, & Beale, 2004) there are a number of common interface styles including *Command line interface*, *Menus and navigation*, *Natural language dialogue*, *Question/answer and query dialog*, *Form-fills and spreadsheets*, *WIMP*, *Point-and-click*, *Three-dimensional interfaces*. These and other styles are often not only styles but they also define the type of user interface (UI). Other examples of styles are: *Direct manipulation style*, *Drag-and-drop*, *Crossing based interfaces* (Choe, Shinohara, Chilana, Dixon, & Wobbrock, 2009), *Gesture interface* (Long, Jr., Landay, & Rowe, 1999), *Motion tracking interface*, *Tangible UI* (Fishkin, 2004), *Voice UI* (Brewster, 1994), *Zoomable UI* (Marinos, Geiger, Schwirten, & Göbel, 2010), etc. Several different styles can be simultaneously present in one user interface; some styles are a specific subset of the other.

In our research we worked and experimented with the most widely spread UI for personal computers - WYSIWYG¹² Graphic User Interface (GUI) with WIMP style, direct manipulation including point-and-click, drag-and-drop style, and the simplest technique of 3D interfaces style, sometimes also with gestures and voice UI. Their brief description follows (Dix, Finlay, Abword, & Beale, 2004):

WIMP (windowing system)

It is the default interface style for the majority of interactive computer systems in use today. It is a GUI based on windows, icons, menus, and a pointing device, typically a mouse¹³ (van Dam, 1997).

Point-and-click

To this category belong interfaces where virtually all actions take only a single selection, e.g., a click of the mouse button, or a touch on the touchscreen. It is represented by highlighted words, maps or iconic buttons in hypermedia, e.g., World Wide Web pages.

Three-dimensional interfaces

Represented by virtual reality, but it also covers the simplest technique where ordinary WIMP elements are given a 3D appearance using shading, e.g., to highlight active areas.

¹² WYSIWYG - an acronym for *what you see is what you get*. The term describes a system in which content displayed during editing appears very similar to the final output.

¹³ Sometimes abbreviated from Window, Icon, Mouse and Pull-down menu

Direct manipulation

This style involves the continuous representation of objects of interest, and rapid, reversible, incremental actions and feedback that correspond at least loosely to the physical world. It is represented by, e.g., resizing a graphical shape such as a rectangle by dragging its corners or edges with a mouse, but this style can be applied also to interfaces for blind or vision-impaired users using a combination of tactile and sonic devices and software.

Drag-and-drop

It is the action of clicking on a virtual object and dragging it to a different location or onto another virtual object to invoke many kinds of actions.

Gesture

It is a movement of the hands, face, or other parts of the body, which can be captured by camera or by an input device, usually a mouse. A gesture can represent a command for the system. It is represented, e.g., by mouse gestures, multi-touch gestures for touchscreens, or face gestures. Gestures are also Pointing, Selection (Point-and-click), Drag-and-drop, Rollover, Menu traversal, and Goal crossing.

Voice user interface

It allows the user to interact through a voice/speech platform in order to initiate an automated service or process.

Each of described styles of interaction is more or less intuitive and all of them have a potential to serve for direct interaction, but it is dependent on the IO devices (see section 3.3).

3.2 Rules for a Good User Interface

We are interested in making control as comfortable as possible. For example to speed up control and make it more intuitive for the user independently of the input devices. There are several things that help designers to design such an interface. At first, there have to be good triggers that represent and can trigger some of the system's available commands. The majority of such triggers are visual, but the system can handle also gestures (usually from a mouse) or spoken commands. Since different types of UIs exist, and not each of them is important for this

work, in this section the WIMP GUI is discussed because our applications run under this type of operating system.

Under GUI belongs everything that is visible on the screen and the user can somehow interact with it. Concerning the current WIMPs, which are modeled as a desktop metaphor, the user can see and interact with windows and icons (using a pointer). These windows contain numerous widgets such as icons, menus, buttons (with tooltips) ordered usually in toolbars or ribbons, dialog boxes, etc.

From Nielsen's set of heuristics (see p. 10) we know that a good interface also needs:

- Help – not only as a documentation, but also as different types of hints, e.g. tooltips
- Visible system status – via status bar, type of pointer or by change of the basic characteristic of the activated object
- Minimize user memory load – use also icons and symbols instead of simple text
- Flexibility and efficiency of use - provide shortcuts and hotkeys, function keys, accesskeys, toolbars, which allow the user to accommodate the application by setting; create automatic adaptation (personalisation)
- Match between system and the real world – allow snapping, aligning and different gestures.

There are many other interface characteristics, but we listed especially those that we dealt with during our research.

3.3 Hardware Supported Interaction

To understand exactly how interaction works and what the pros and cons of input and output devices are, i.e. what is intuitive about them and what is not or how they restrict users, it is necessary to look at them more closely. Special attention is paid to standard devices, since these are exactly our target devices.

Keyboard

Many computer-like devices have a keyboard. It is an inseparable part of our personal computers, and together with the mouse, a routine input device. There are different types of keyboards, but most of them have about 100 keys usually in 5 rows. For special interaction needs such as shortcuts, the keys Shift, Alt and Ctrl are used in combination with other keys. There is no clear parallel in the real word to the keyboard => it is unintuitive.



Mouse

A basic type of mouse has two buttons and one wheel. There also exist many other types of mice with more buttons to which users can map different functions, but they are not in our focus. Mice are very valuable devices that, unlike keyboards, are very close to a human type of interaction: people are used to pointing to things with their index finger and to the same finger (**left button** of mouse) is mapped **selection** as an indirect kind of control. Only direct control is more intuitive, which is present in, e.g., touch screens, but mice have a higher degree of precision. The second valuable mouse property is **mouse movements**, consistently mapped to **pointer movements**. It is again, indirect control, but still intuitive. If there is a 3D workspace, mouse movements can be mapped differently, which is described in detail in section 4.3.1 on page 48.



The next mouse function is **right click** (done by the middle finger). It is usually mapped to a **context menu**, which is connected to selection and it often simplifies the user's interaction. There is no parallel with right click in the real world (= unintuitive), so it has to be learned, but it is also a standard, so many users already know this feature.

The mouse **wheel** usually serves for moving focus within an object of interest. The most common implementation is moving an object under the pointer, but sometimes it can move the pointer itself, e.g., within a menu. Working within a 3D space, the wheel is often used for zooming in and zooming out. The intuition of this indirect control can be found in moving / zooming an object under the pointer, resembling turning the wheel on the mouse. Similar, but with direct control (= more intuitive) can be found, e.g., on iPhones the UIPickerView element, which is a space saving version of combo-box for a small touchscreen. It looks as a wheel and can be turned by the index finger.

Monitor

Whereas the human sight is the best sense, a monitor (as an output device representing the seeing/vision modality) has a very important role in the interaction. Again, as well as with the keyboard and the mouse, it is important to make things as close as possible to the reality and if it is not possible, it should be the most intuitive. This means that every (often used) function and reaction of an application has to be easy to find, easy to determine, easy to understand. For this purpose there are GUI elements such as menus, toolbars or ribbons with icon-buttons, status bars and so on. These are usual and



standardized ways to offer the user the control he/she needs (more about UIs in section 3.1 Interaction Styles). Sometimes, it is not enough and different companies design their own new way of, e.g., object behavior, which allows the user to interact with it easier (it means user performance is quicker and learnability of the system remains acceptable).

In our case, the most intuitive screen would be a touch screen, but that is not common for PCs. To resemble the real world, the first step is to use perspective imaging for our 3D scene. The second step is to offer the user such a view that would remove the deficiencies of a 2D screen. What these deficiencies are and how to remove them is my main area of research, so a detailed description follows in later sections and chapters.

Web-cam, Microphone and Loudspeakers



Besides the keyboard, the mouse and the monitor, there are other standard IO devices, such as a web-cam, a microphone, and loudspeakers. Since there is no guarantee that the user has them, the majority of applications do not count on them, because (to keep high usability - effectiveness) they should work properly for any type of user – also the one without these devices.

An example of interaction via a camera connected to a standard PC is a user tracked by a camera and according to his/her head or eye movements an object is moved on a screen or a scene itself. Other example of interaction is an audio signal (spoken word) captured by a microphone, recognized by a speech recognition algorithm and the pronounced command performed. Loudspeakers can, via speech synthesis, announce whatever is necessary. Omitting the fact that not every user owns these 3 devices, it will take some time to enable computers to interact through them in a way that is intuitive for people. Presently systems working with these advanced technologies require the user to learn new interaction habits which affect the learnability of the system. E.g., the user needs to pronounce correctly or needs to train a speech recognition system for his/her own type of pronunciation. After that the user needs to learn a set of commands that can be spoken. In case of visual commands captured by a camera, the user needs to learn which movement means which command.

We experimented with interaction through all of these basic devices to provide users easier and more intuitive interaction.

3.3.1 Other Input and Output Devices

In addition to standard IO devices there exists a plethora of other devices which try to serve the user in a more natural way, the way he/she is used to interact in the real world. In the following lines there is a categorized list of them. We divided them into groups depending on the part of the user's body that they relate to:

- Pointing devices for **hands and fingers**: joystick, trackball, various space navigators, gloves and other different types of haptic devices, various keyboards, game-controller, gamepad, game-paddle etc.
- Pointing devices for the **index finger**: touchscreens, multi-touch screens, touchpads and pointing stick, tablets, different types of pens, etc.
- Devices for **hands or legs**: steering (racing) wheel, pedals, light gun, dance pad, etc.
- Devices for **eyes** (head) are different types of:
 - **glasses** (for stereoscopic displays): anaglyphic, polarized, semitransparent, liquid crystal shutter glasses
 - **displays** and projectors: head mounted display, holographic display, volumetric display, 3D display, etc.
 - (printers and other devices)
 - **environment**: CAVE (can also require glasses)
- Motion tracking devices, usually camera like devices, tracking a **body part's position**: track reflex or illuminating points (or magnets tracked by magnetic system), track color patterns or simply send any taken picture to recognition algorithms and recognizing various actions and movements, e.g., whether the user is standing, moving, sitting, waving, smiling, touching triggers, etc.
 - Motion sensing devices: wii remote, play station move, Kinect or any device with a built-in accelerometer.
- Position tracking devices detecting the **user's position**: GPS detects position on the Earth and with the combination of a camera image can detect even the surroundings; any device with a built-in gyroscope can measure orientation toward the Earth surface.

Concerning devices for hands or for the index finger, there are some that are better designed than others for human natural

interaction. They include pens and touchscreens that are sensitive also to **pressure** and are able to distinguish different levels of it. Since people have a pattern of pushing either strongly or lightly, often a matter of their emotions or personality, this type of device can be used for signature authentication, among other things.

We named only a few examples. There are more, but for the purpose of our work it is not necessary to go into greater detail.

4. Current Interaction Approaches

The following three sections describe related work connected to all of our three goals, each section for one goal.

4.1 Remote Object Interaction

Our first goal was to propose and verify a method of a computer aided interaction with remote objects in an environment (off the computer screen) using common and affordable equipment (including input and output devices). The problem of pointing at remote objects, as an interaction problem, can be understood at different levels:

1. Indoor-indoor: The area of interest is remote, but the user has a camera there and Internet connection can bring him/her a picture of it. Interacting with this picture the user can interact with objects within the remote area.
2. Indoor/table: The user is very close to the area of his/her interest, but it is too big/wide. The user cannot reach each corner of the area by simple hand stretching.
3. Indoor/room: The user cannot reach the area of his/her interest by hand but can use a laser pointer.
4. Outdoor: The user cannot mark the object of interest even by laser pointer, because the object is too far away.

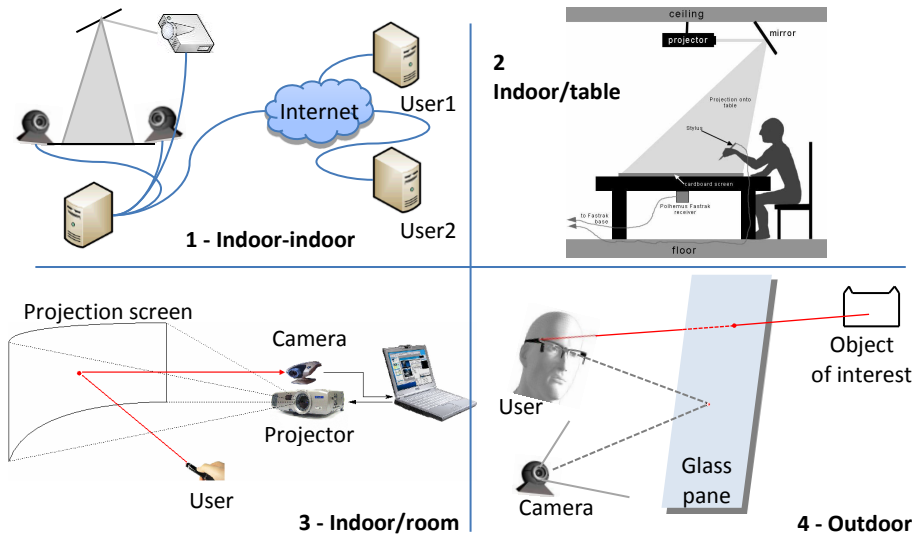


Figure 4: Four categories of pointing at remote object

In the following subsections several solutions from each group are briefly introduced with a focus on interaction and devices needed to execute and process its execution.

4.1.1 Interaction from Indoor to Remote Indoor

One of the solutions, where a picture taken by a camera is brought from a remote area to the user's screen through the web and in the same way he/she can interact with it, is the Ishihara and Ishihara study - Remote direct pointing system using graycode (Ishihara & Ishihara, 2006). The authors apply gray-code to a remote direct pointing system. Gray-code is a method for automatic projection calibration. Gray-code binary patterns are projected to discover the locations of objects within the projector's perspective. In addition to this main feature, gray-code is capable of identifying a location within the projector's perspective from wherever the gray-code binary patterns can be seen. The authors take advantage this of gray-code to build a remote direct pointing system. They built a prototype (see Figure 5) of the system that helps remote users draw directly onto remote objects. In the prototype, users see remote objects through cameras and draw on the objects simply by positioning the pointer on the images from the cameras. This property helps remote users get involved in remote environments. We describe the design of the prototype and also show an

example of the prototype in use. The remote pen enables remote users to draw directly onto a remote desk or note.



Figure 5: The setup of the prototype for Remote pen (left). Student's camera view, where is shown how he draws a diagram representing the direction of the outer product and the magnitude (right) (Ishihara & Ishihara, 2006)

The prototype of a remote direct pointing system using graycode includes a projector, two CCD cameras, and four computers (two of them perform two Cam# views, one performs Camera switch, and the other performs Gray-code manager, Gray-code generator, and Application). All pieces of equipment are networked by 100Mbps Ethernet. The Remote pen enables remote users to draw directly onto a remote desk or note.

4.1.2 Indoor Interaction within Table Distances

With magnification of touch displays (often tabletop displays) as working areas, the user cannot reach the far side of a display by a direct input device such as a stylus. The distance between the object of interest and the user can be about one meter. The two following examples propose an extension of the original interaction method to cope with unreachable area.

TractorBeam

Parker et al. (Parker, Mandryk, & Inkpen, 2005) proposed augmenting a stylus to allow remote pointing. Results from their work demonstrate that remote pointing is faster than stylus touch input for large targets, slower for small distant targets, and comparable in all other cases. They found, when given a choice, people utilized the pointing interaction technique more often than stylus touch. Based on these results they developed the TractorBeam, a hybrid point-touch input technique that allows users to seamlessly reach distant objects on tabletop displays.

The hardware setup for the top-projected tabletop display that was used consisted of a ceiling-mounted projector, mirror, desktop PC, and wooden table. The output from the PC was projected onto the mirror, which reflected the image onto the table (Figure 6). Input was received via a tethered stylus and receiver attached to a Polhemus Fastrak (six degrees of freedom 3D tracking system).

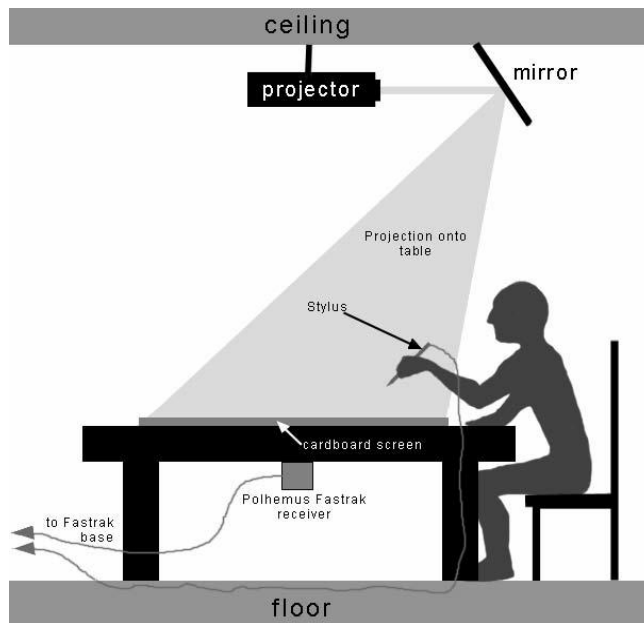


Figure 6: Top-projected table hardware configuration
(Parker, Mandryk, & Inkpen, 2005)

A multi-directional task (2D Fitts discrete task) was used under three conditions:

- Touch – Participants selected objects by touching the stylus to an item on the table
- Point – Users selected objects by pointing at them with a stylus (using it like a laser pointer, with a cursor appearing on the table).
- Reach-and-point – Users selected objects by pointing at them (similar to the point condition) but were encouraged to reach out over the display to reduce the distance between stylus and target

The Vacuum: Facilitating the Manipulation of Distant Objects

The Vacuum (Bezerianos & Balakrishnan, 2005) is a new interaction technique that enables quick access to items on areas of a large display that are difficult for a user to reach without significant physical movement. The vacuum is a circular widget (bull's eye) with a

user controllable arc of influence that is centered at the widget's point of invocation and spans out to the edges of the display (see Figure 7). Far away objects residing inside this influence arc are brought closer to the widget's center in the form of proxies that can be manipulated in lieu of the original. Authors conducted two experiments which compare the vacuum to direct picking and an existing technique called drag-and-pick. Their results show that the vacuum outperforms existing techniques when selecting multiple targets in a sequence, performs similarly to existing techniques when selecting single targets located moderately far away, and slightly worse with single targets located very far away in the presence of distracter targets along the path.

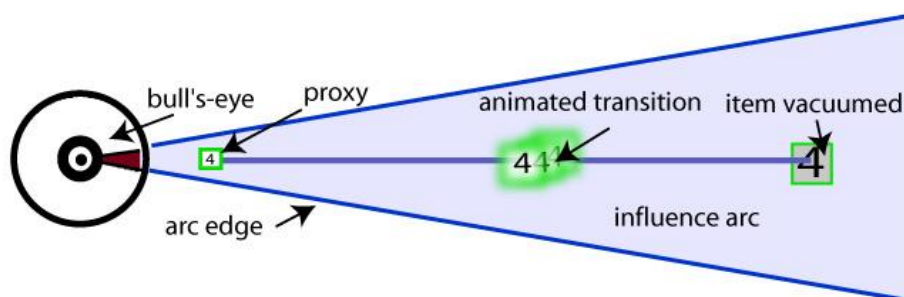


Figure 7: Vacuum (Bezerianos & Balakrishnan, 2005)

The authors used a 16' wide, 6' high, back projected large display, with imagery generated by 18 projectors (1024x768 resolution each) in a 6x3 tiling for an effective resolution of 6144x2304 pixels. The projectors are driven by a cluster of 18 workstations. A camera-based Vicon motion tracking system (www.vicon.com) tracked a pen's movement over the screen. Although the system could track the pen in 3D space, the authors used only x-y screen movements, a 5 inch hover zone, and a single button

4.1.3 Laser Pointer

The next level of remote area is when a user cannot reach the area of his/her interest by hand but uses a laser pointer. The distance between the object of interest and the user can be several meters. These solutions are connected to big screens e.g. wall displays. In 2001 Olsen and Nielsen T. described a technique using a laser pointer and a camera to enable people to interact at a distance from a display surface, e.g., group meetings and other non-desk situations. They described calibration techniques (to synchronize the display and camera coordinates) and a series of interactive techniques for navigation and

different types of entry (Olsen & Nielsen, 2001). These interaction techniques consist of five events, which the user can evoke by laser:

- LaserOn (X,Y)
- LaserOff(X,Y)
- LaserExtendedOff(X,Y)
- LaserMove(X,Y)
- LaserDwell(X,Y)

These techniques were later used and enhanced for different specific situations; five of them are briefly introduced in the following paragraphs.

An Independent and Non-Intrusive Laser Pointer Environment Control Device System

A laser pointer can be used to control different things. One is the control of a computer environment for the handicapped. The system designed by Chávez et al. looks for a laser spot position which is projected on the environment by using a laser pointer. Handicapped people can thus select the device they want by using the laser pointer. Once the laser spot is found, the device is controlled by means of a domotic system, using KNX architecture. The system is able to recognize and act on the device that the handicapped person wants to use (Chávez, Vega, Olague, & Montero, 2008).

The video camera used by the system is a household video camera with 800.000 pixels in the CCD. The laser pointer used is a laser pointer class II with maximum power < 1mW and a wave length of 630-680 nm.

Laser Pointer Tracking in Projector-Augmented Architectural Environments

Kurz et al. presented a system that employs a custom-built pan-tilt-zoom camera for laser pointer tracking in arbitrary real environments (Kurz, Hantsch, Grobe, Schiewe, & Bimber, 2007). Once placed in a room, it carries out a fully automatic selfregistration, registrations of projectors, and sampling of surface parameters, such as geometry and reflectivity. After these steps, it can be used for tracking a laser spot on the surface as well as an LED marker in 3D space, using inter-playing fish-eye context and controllable detail cameras.

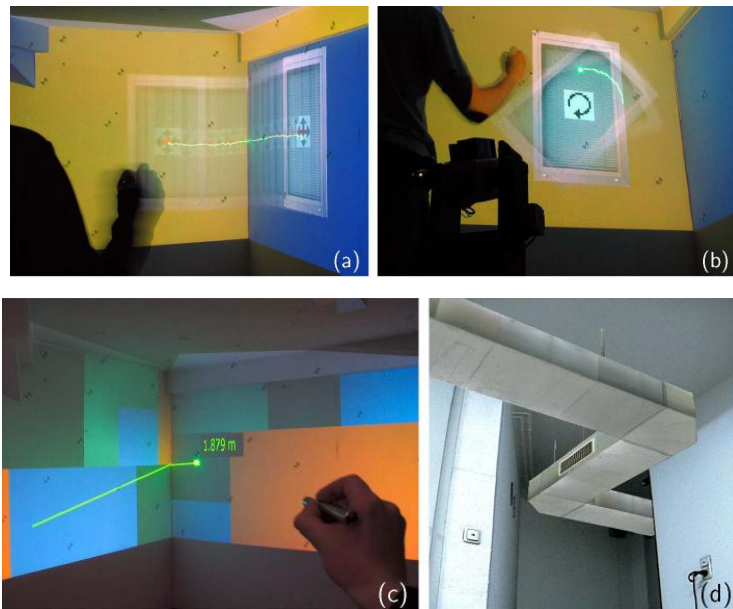


Figure 8: Basic object manipulation techniques such as translation (a) and rotation (b) are illustrated in long exposure photographs. Augmentation can be projector-based (a-c) or via video-see-through (d). These application examples show basic augmentations of building structures (a,b,d), distance measurements (c) and material-color simulations (c).
(Kurz, Hantsch, Grobe, Schiewe, & Bimber, 2007)

The captured surface information can be used for masking out areas that are problematic for laser pointer tracking, and for guiding geometric and radiometric image correction techniques that enable a projector-based augmentation on arbitrary surfaces (Figure 8).

The system, as a distributed software framework, couples laser pointer tracking for interaction, projector-based augmented reality (AR) as well as video see-through AR for visualizations, with the domain specific functionality of existing desktop tools for architectural planning, simulation and building surveying.

The hardware configuration of this system consists of two video-cameras - a low resolution wide angle context camera and a high resolution PTZ (Pan tilt zoom) detail camera. A micro-controller and two stepper-motors with their controllers are utilized to rotate the detail camera and an attached laser module (for detecting distances). Both cameras are directly connected to a PC that controls motors, camera settings, and the laser module.

Vision based laser pointer interaction for flexible screens

Kim et al. present new interaction techniques that use a laser pointer to directly interact with a display on a large screen (Kim, Lee, Lee, & Lee, 2007), which can be very useful during group meetings and other non-desk situations where people should be able to interact at a distance from a display surface.

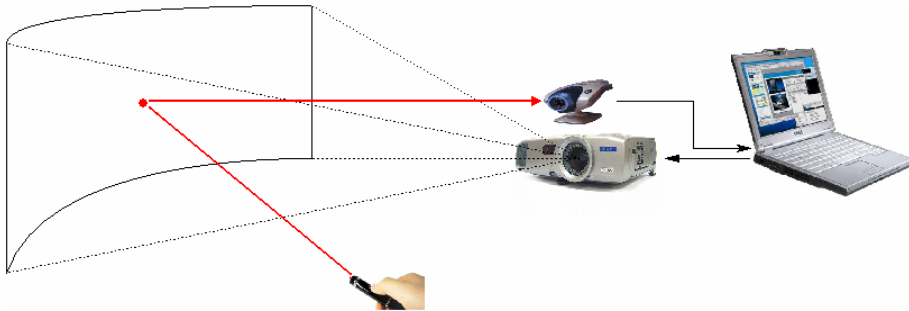


Figure 9: The vision based interaction system; The portable LCD projector and USB2.0 camera are placed at the front of the screen, while the user can control the mouse function using a laser pointer (Kim, Lee, Lee, & Lee, 2007)

The camera is subsequently used to detect the position of the pointing device (such as a laser pointer dot) on the screen, allowing the laser pointer to emulate the pointing actions of the mouse. The laser pointer will behave as an active point on the projected display where the user can interact. This vision-based system is augmented with a natural interface that enables the user to interactively refine the suggested rectification. This makes it very easy for users to execute fast and continuous commands. The interaction model developed behaves like a "smart interaction system." The vision based interaction system requires no special hardware and runs on a standard computer.

Laser pointer interaction techniques using peripheral areas of screens

Shizuki et al. presented new interaction techniques that use a laser pointer to directly manipulate applications displayed on a large screen (Shizuki, Hisamatsu, Takahashi, & Tanaka, 2006). The techniques are based on goal crossing, and the key is that the goals of crossing are the four peripheral screen areas, which are extremely large. This makes it very easy for users to execute commands, and the crossing-based interaction enables users to execute fast and continuous commands.

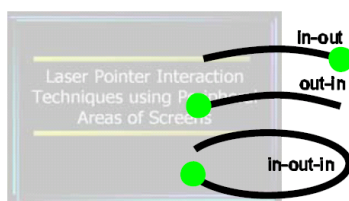


Figure 10: Three types of basic crossing
(Shizuki, Hisamatsu, Takahashi, & Tanaka, 2006)

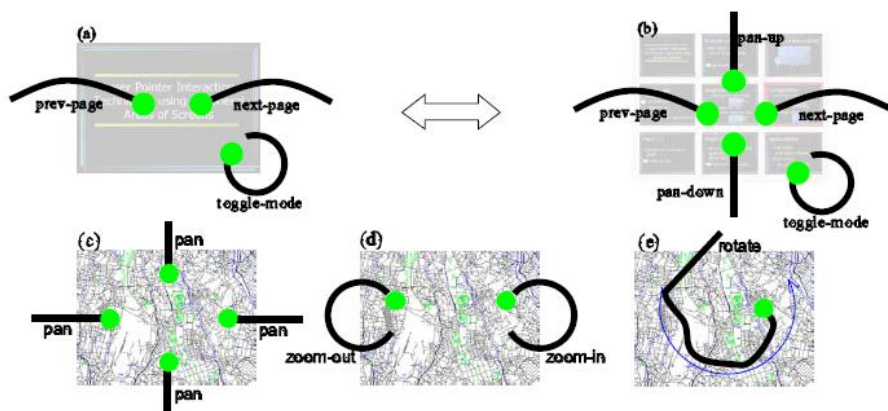


Figure 11: Crossing-command mappings for a slideshow and a map viewer application (Shizuki, Hisamatsu, Takahashi, & Tanaka, 2006)

The system consists of a laser pointer with an on/off button, a USB camera to capture the movement of the laser spot projected on the screen, and a presenter's computer. The presenter can use the laser pointer to control projected software by moving the laser spot and turning the beam on or off. He/she can control the slideshow using the laser pointer, avoiding the necessity of returning to the computer, which may be positioned far away.

A Practical System for Laser Pointer Interaction on Large Displays

While the previous works explain well the different systems for the laser pointer as an interaction device, whose dot location is extracted (and used as a cursor position) from an image of the display captured by the camera, Ahlborn et al. focused on some important practical concerns such as the design of a tracking system and key practical implementation details. They presented a robust and efficient dot detection algorithm that allows us to use their system under a variety of lighting conditions, and to reduce the amount of image parsing required to find a laser position by an order of magnitude (Ahlborn, Thompson, Kreylos, Hamann, & Staadt, 2005).

Their system contains a PC, a display system of a 3×2 tiled display with a total resolution of 3072×1536 pixels, four Canon VC-C4 cameras (each with a resolution of 720×480) attached to WinTV frame grabbers, the laser pointer (class IIIa, red). A Linux Infra-Red Controller (LIRC) device is used for detection of button presses.

4.1.4 Outdoor Interaction

Outdoor interaction can be understood as an interaction with objects within an outdoor environment. There are plenty of interaction possibilities for interacting with virtual environments or small environments (as large as a room). Most of them belong to the category of Virtual Reality or Augmented Reality and although the user can interact with objects within such an environment or even though it looks like an outdoor space, it is still indoors. Moving to a real outdoor space, one can find different devices (mostly cell phones, see Figure 12) that can handle its orientation or the user's GPS position or at least somehow communicate with other devices around.



Figure 12: An example of applications for outdoor star observation: Google Sky Map¹⁴ (left) and Star Walk¹⁵ (right), both screenshots with inverted colors

None of these solutions fits our goal, because they significantly increase the final price. Therefore we were interested in other possibilities, which help the user by giving additional information on observed objects that are hundreds of meters distant or even further.

¹⁴ <http://www.google.com/mobile/skymap/>

¹⁵ <http://vitotechnology.com/star-walk.html>

Outdoor remote object observation

There is an old method, which provides the observer with an enriched view of the world around and which requires neither a computer nor any other electronic device. Only thing needed is a well placed glass pane with specific information. An example of using such a glass pane, which gives a better view of the area distanced just a few meters from the observer, is a pane depicting the parts of the ruins, which are absent. Thus, observers can have a very specific idea how a given object looked in the times of its glory (Figure 13).



Figure 13: An illustration of augmented reality without the use of computing resources, only through a glass pane – a terrain with ruins and the glass pane with ruins' complement (left); observer's view through the glass pane (right)¹⁶

Another example can be found in the field of glass sundials (Figure 14). Their history started in 1529. Sundials can accurately tell the dates of the solstices and equinoxes and can have date lines for birthdays or anniversaries, too.

These examples show that glass has been used for centuries to give the observer information in which he/she is interested. A glass pane is used even today as a tool for observers in different situations dealing with long distances.

¹⁶ Pictures adapted from http://www.visionpubl.com/past_present_show.asp?pag=6&id=6



Figure 14: Spectra sundial (21st century) (Carmichael, 2011)

4.1.5 Summary

All of these solutions work with different types of “remote object scenarios” and all of them have in common the need for the localization detection of the user’s pointer, whether it is a virtual (projected) pointer within a real environment or real pointer within a virtual (projected) environment. In all of these solutions, the calculation was based on data from the camera. The larger distances require not only the camera but also a laser pointer. The five mentioned solutions in the laser pointer group use the laser to point at objects several meters distant. This allows system designers to work with a laser dot on the surface. The dot (point) coordinates are detected using one or more cameras, but prior calibration is necessary. If the area is bounded, it creates space for more specified interaction events. The same problem - short distances - exists within virtual environments or with small environments.

In our case we deal with an outdoor environment, where there is no surface – objects of interest are too far and the area does not have bounds. This is the reason, why none of the indoor solutions using a laser pointer are suitable for our scenario. Other solutions that are suitable to work with long distances (tens of meters or longer) require different hardware, which does not satisfy our initial specification, (especially affordability). Finally, there is another method using only a glass plate (with static image) to enhance the user’s observation, which works for distances from meters to an astronomic unit or even longer. This inspires us to use the glass plate also in our solution.

4.2 Lowering the Demands on the User's Interaction

Our second goal was to propose and verify a method, which stores the user's actions and thus reduces the demands on his/her interaction when retrieving web information. This is a form of adaptable user interface.

It can be found in operating systems. For example, the Microsoft Windows Start menu offers in quick choices the most used and most recently used applications. Other similar arrangements can be found in many applications, where menus are shortened to only those items that have been already selected in the past. This problem, however, is not sufficiently addressed in the field of information retrieval from the Internet, so we focused on lowering demands when the user repetitively looks for specific information in a specified field. It means the user knows where to search for and how to filter the information available at the location – his/her favorite source. In other words, in our case the user is able to formulate his/her requirements in greater details and can be explicit. Examples of such requirements could be: "I want to monitor this specific list of stocks on the stock-market and I have no interest in the fluctuation of other stocks or of the general index." Or, "I need to have the current weather forecast for the city where I live and I prefer to have it in textual and image form." Since this is his/her known area and the experienced user knows where and how to find (manually) the information he/she is interested in, the remaining problem is: How to transform such a requirement into a computer language so that a computer can look for the information automatically (instead of the user).

To understand this problem practically, let's have a user, who looks for information on bus departures from his/her home to his/her work. It takes a little bit of time till he/she opens the relevant web page in his/her web browser – it always takes him/her a few seconds to perform this task, which consists of several interaction steps. The time depends on the degree to which the user is capable of customizing the system he/she is working with and also on how much different settings allow him/her to speed up obtaining the desired information. Some time can be saved by cookies, since they remember the last choices and represent a system adaptation.

The user's requirement can be formulated for example like this: "I want to know when my bus is going from where I am now and in the

usual direction." It is important to notice the words "my," "where" and "usual," because these assume an application is able to estimate his/her choice of bus number, where he/she is and which direction he/she wants to travel.

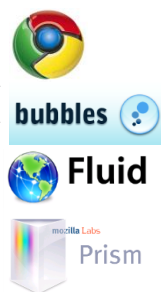
The following subsections describe various ways that help the user to quickly obtain desired information from the Internet (by means of reducing the number of interaction steps). We start with a description of classic web browsers and their possibilities and finish with widgets and gadgets. We do not deal with highly specialized web-applications.

4.2.1 Fast Information Retrieval Through Web Browsers



The most favorite way of retrieving information from the web is the well-known browsing. For this purpose there are **web browsers** e.g. Microsoft Internet Explorer, Mozilla Firefox, Safari or Opera. Some ways the average web browser can save time are to set up some settings e.g., to save his/her favorite web page via "Add to Favorites," to make some page as his/her "home-page," to "Show the windows and tabs from the last time" when the web browser starts. Such settings allow the user to set up different options about web pages, but there is no possibility to specify or to ask for specific information within the web page (if the user wants just a part of the page). Microsoft Internet Explorer 8 brought a little improvement with Web Slices (Microsoft Corporation, 2011), which use simple HTML markup to represent a clipping of a web page, enabling users to subscribe to content directly within a web page.

Moving from generic web browsers, other type of acceleration can be found in **site specific browsers**. This type of browser is designed to create a more comfortable environment for the user, especially when browsing "the favorite" sites e.g. for e-mails or on different types of social networks. Examples of the site specific browsers are: Fluid (for MacOS X), Mozilla Prism, Google Chrome or Bubbles. These are web-applications, which have the same core as web browsers but from the outside they look like desktop applications. They offer the drag & drop function and have many other features, they may have some settings, which can be manually set up and then the user obtains his/her information even quicker as in a web browser, but they still do not guess the user's focus, do not give a chance to filter (specify which part of which web page) and do not offer the option to choose a way of presentation.



Apart from bookmarking systems built-in to web browsers, users can take advantage of **bookmarking web services**, such as the social bookmarking system Delicious (Yahoo!, 2011), formerly del.icio.us. Such



services provide them with the possibility to organize their bookmarks by using tags and to have their bookmarks available independently of the user's location and browser.

Another option, which can significantly speed-up user access to relevant information are **personalized and adaptive web-based systems** (Brusilovsky & Millán, 2007), especially when combined with site-specific browsers. An appropriately trained personalized web based system can often display the information the user is looking for directly on the first page. Such personalization and adaptation is obtained via well-designed user models.

The **user model** is a reflection of users' understanding of a product. It includes how they perceive the objects and activities involved in the operation of the product (Shubin, 1999). It contains all information that the system knows about the user. It is generally initialized either with default values or by querying the user and then maintained=kept updated by the system. Users may be able to review and edit their profile. User actions and events at various conceptual levels, such as mouse clicks, task completion and requests for help, are analyzed - an analysis engine combines the user model with other models of the system to derive new "facts" about the user and update the user model with the derived facts (or initiate an action in the application such as interrupting the user with a suggestion). The analysis engine also responds to queries from the application (Kules, 2000).

4.2.2 Widgets and Gadgets

Without implementing its own engine or a robust system, a chance for reducing interaction steps can be found in **widgets** (sometimes also called *gadgets*) or **mashups**.

Mashups are services that combine two or more information sources to create a new service. Tuchinda et al. specify four tasks in the construction of mashups (Tuchinda, Szekely, & Knoblock, 2008):

- (i) data retrieval
- (ii) source modeling
- (iii) data cleaning
- (iv) data integration

Mashups are currently widely researched, and much research is being carried out, especially in order to facilitate the creation of mashups (Wong & Hong, 2007).

A widget, on the other hand, encapsulates a single information source and is responsible for the three first tasks (i, ii and iii). Thus widgets are not some elements that help the user to navigate or to orientate or to pick a choice, but they are single-purpose mini-(web-)applications, which typically have a minimal size and are dedicated to bring a simple solution based effect while a user is working with a computer. Widget functionality is oriented to one specific goal – to display very specific information. They can be of two types, either for the web (web-widgets) or for the desktop (widgets) (Caceres, 2008). The latter ones can be used for computer as well as mobile devices. Research has already been done on their usability (Han & Park, 2009), their adaptive managing (Boström, et al., 2008) and the web widgets were used also to enable the semantic web (Mäkelä, et al., 2007).

We focused on the desktop widget for computers, which can be freely located and easily combined within the desktop. The most often used engines for widgets or gadgets are:

- *Konfabulator* (Yahoo!, 2011) from Yahoo! for Windows XP+ and MacOS known as Yahoo! widgets (examples on Figure 15 (d))
- *Windows Desktop Gadgets* (Microsoft, 2011) (called *Windows Sidebar* in Windows Vista) from Microsoft for Windows with gadgets on the Windows desktop (examples in Figure 15 (a))
- *Google Desktop Gadgets* (Google, 2009) from Google for Windows XP+ in a form of Google Desktop (examples in Figure 15 (b))
- *Opera Widgets* (Opera Software, 2010) from Opera for Beta MacOS 10.5 and Windows XP+ (examples in Figure 15 (c))
- *Dashboard* (Apple, 2011) (for web *WebKit*¹⁷) from Apple for MacOS X as the 2nd desktop with widgets (examples in Figure 15 (e))

Most of them use a kind of API which processes mainly HTML, XHTML or XML and CSS files plus JavaScript, sometimes Perl, Python, Ruby or C++. There are not major differences between the visuals (see Figure 15).



¹⁷ WebKit, <http://www.webkit.org>, also for S60 OS



Figure 15: Examples of desktop widgets screenshots taken from web: Windows Gadgets (a), Google Desktop Gadgets¹⁸ (b), Opera Widgets¹⁹ (c), Yahoo! Widgets²⁰ (d), and Dashboard²¹ (e)

There are some differences between desktop widgets and gadgets made in different engines. From the user perspective some are represented by views or icons which are located in a standard sidebar of the desktop and they become active only after click initiation where the icon spreads itself to the desktop. After this the widget can be relocated as wished (sometimes the relocation is limited within the sidebar). Some have almost double sized sidebar wideness and provide the service during the whole time they are active. When they are clicked, they spread themselves and increase the service quality or quantity.

From the implementation point of view there are three possibilities how a user can get his/her own personal widgets. The user needs first to decide which API he/she wants to use according to his/her operating system (see Table 3) and software he/she has already installed - if it is not a part of his/her system, he/she needs to install it. Then there are three choices on how to create a personalized desktop enhanced by individual widgets:

¹⁸ Adapted from http://desktop.google.com/images/sidebar_gd55_intl.jpg

¹⁹ Adapted from <http://files.myopera.com/DnSee/files/Desktop1.JPG>

²⁰ Adapted from <http://img442.imageshack.us/img442/6848/mine23it4.jpg>

²¹ Adapted from http://developer.apple.com/library/mac/documentation/userexperience/conceptual/applehighguidelines/art/mt_dashboardview.jpg

1. To find a widget on the web page with plenty of ready-to-use widgets, download it, manually set up it and use it.
2. To read a tutorial for extending a generic widget and follow simple instructions to create a specific one (here comes into the consideration also the programming language – see Table 3).
3. To read a tutorial for developers and program his/her own widget.

What option will the user choose depends on the type of information that is expected to be displayed (the way of displaying is not taken into account for now).

Table 3: Comparison of different widget engines according to the operating system they run and language they can be programmed in.

Engine	OS support			Progr. language support				
	Windows	Linux	Mac OS X	JavaScript	Perl	Python	Ruby	C++
adesklets	✗	✓	✗	✗	✓	✓	✗	✗
Adobe AIR	✓	✓	✓	✓	✗	✗	✗	✗
AveDesk	✓	✗	✗	✓	✗	✗	✗	✓
Dashboard	✗	✗	✓	✓	✓	✓	✓	✓
DesktopX	✓	✗	✗	✓	✓	✓	✗	✓
gDesklets	✗	✓	✗	✗	✗	✓	✗	✗
Google Desktop Gadgets	✓	✓	✓	✓	✗	✗	✗	✗
Kapsules	✓	✗	✗	✗	✓	✓	✗	✗
KlipFolio	✓	✗	✗	✓	✗	✗	✗	✗
Microsoft Gadgets	✓	✗	✗	✓	✗	✓	✓	✓
Opera	✓	✓	✓	✓	✗	✗	✗	✗
Plasma	✓	✓	✓	✓	✓	✓	✓	✓
Screenlets	✗	✓	✗	✗	✗	✓	✗	✗
Serious Samurize	✓	✗	✗	✓	✓	✓	✓	✓
SuperKaramba	✗	✓	✗	✓	✗	✓	✓	✓
WebKit	✓	✗	✓	✓	✗	✗	✗	✓
Yahoo! Widgets	✓	✗	✓	✓	✗	✗	✗	✗

4.2.3 Summary

Just like the site specific browser, the ready-to-use widgets meet the demand of the majority of Internet users. Therefore nonstandard

requirements are not covered by the first choice (existence of already implemented widget). If there is already a service as an RSS²² or a web-service, which can be requested for information, extending a generic widget is sometimes enough. But in case of non-existent ready-to-use widget or service, the only choice is to program it. This last one also gives a space for the developer to implement some features, which can offer the user some kind of automatic adaptation or personalization. But generally there is no effort to implement widgets for one purpose with a broad usage (i.e. independent of information source); those which obtain information from the Internet all are exactly one site or exactly one web-service oriented.

In general, obtaining information from the Internet is not personalized and users have to manually set up the system or hope for the web site with implemented harmless cookies.

²² Really Simple Syndication - a family of web feed formats used to publish frequently updated works

4.3 Interaction in 3D Environments Using a Typical PC

Our third goal was to verify whether different methods of visualized information increases usability of 3D graphical editors, with emphasis on graphical hint for a hovered object within a graphic editor, where this suggestion (hint) visualizes the consequence of object's selection.

To understand the systems providing different functionalities of various 3D worlds, we analyzed them with a focus on their common features as well as their differences. At first, it has to be noticed that there are two basic different control concepts of the 3D world. The first one is when the user is a center of “the world,” where most of the interaction is oriented to this “**center**” and there are very limited possibilities to interact with the world around because it is already set. This concept is mainly used in 3D games, often as first-person shooter (FPS) game genre. The opposite concept, the second one, is when the user is a creator of the world, so the main interaction is oriented “**outside**” to the world. This second type can be found in each 3D graphic editor. According to a chosen type of control concept we can find a typical set of allowed interactions. For our purpose the second type is more interesting, but still the first type cannot be overlooked because there are far more users who already know how to control a 3D game than a 3D editor. These users have already some control habits and expect the same in every other software.

In the following two subsections we closely look at the first and the second control concept to find out what types of control were already used, are currently used and which of their characteristics fit for our purpose. The third subsection contains a description of controls in existing educational software, which is actually the closest related work. This section ends with a summary.

4.3.1 3D games

There are different types of 3D real-time rendering computer games²³. Together with increasing computing power it was possible to implement a wider functionality and richer interactivity. This historical development divides games into those that simulate a 3D world, but a

²³ Using 3D engines; providing 3D world with 3D objects

player can move only on the surface of this world and those where the player is able to move in all 6 degrees of freedom.

3D world + 2D movements

The most widespread games of this type are first person shooter (FPS) games. Such games require the player to only control two axes and their heading – the player moves upon the earth (in buildings, in the hills). The first famous games of this type were Wolfenstein (Id Software LLC, 2002) introduced in 1992 and Doom (Id Software, 1993) introduced in 1993. It was possible to control the game only by one input device, e.g., keyboard, mouse or joystick (see Table 4), but many players very often chose a keyboard (W, A, S, D, Space bar, Ctrl, Alt, Shift) in combination with mouse movements.

Table 4: The default controls in Doom for the most often-used functions²⁴

Function	Keyboard	Mouse	Joystick
Turn right	right arrow	mouse right	joy right
Turn left	left arrow	mouse left	joy left
Move forward	up arrow / W	mouse up or button 2	joy up
Move backward	down arrow / S	mouse down	joy down
Strafe left	(Alt + left arrow) / comma / A	(Alt or button 3) + mouse left	
Strafe right	(Alt + right arrow) / period / D	(Alt or button 3) + mouse right	
Fire weapon	Ctrl	button 1	button 1
Use (open)	Space bar	double-click button 2 or double-click button 3	button 2
Strafe	Alt	button 3	button 3
Run ³	Shift		button 4

Later games have been enriched by look upward and downward, fly upward or jump (A key), fly downward or squat (Z key). The first famous game of this type was Duke Nukem 3D (3D Realms, 1996) introduced in 1996.

3D world + 3D movements

Real 3D game demands that players keep their sense of orientation in a fully 3D environment with a flight model featuring six degrees of freedom (6DoF) in zero-gravity. By employing six degrees of

²⁴ Adapted from <http://doom.wikia.com/wiki/Controls>

movement, the player is given additional control for vertical movement and banking, introducing more movement controls than traditional FPS games. The first famous game of this type was *Descent* (Interplay, 1995) introduced in 1995. This game requires the player (in a spaceship) to navigate labyrinthine mines while fighting virus-infected robots. The control is very similar to *Doom* (Id Software, 1993) - while some players might opt for a well-stocked joystick, others move for the mouse and keyboard combination. The mouse is used to position ship's viewpoint left, right, up, and down. The two mouse buttons usually fire primary and secondary weapons. As for the keyboard portion of this combo, players tend to gravitate to the WASD combination - **W** moves forward, **S** moves backward, **A** strafes left, and **D** strafes right. So far, this set up is like a simple FPS, but several other keys are: rolling left or right the Q and E keys, for strafing up and down players used to choose R and F, as they are adjacent to the movement keys.

Even with the advent of mouse and keyboard combos in FPSs, some players still prefer the keyboard only - a set of four keys (usually the arrow keys) for the ship's viewpoint and another set of four (usually WASD combo) for moving forward, backward, and strafing left and right.

In present 3D games dominate those that offer only 2D movements - on the surface. The demands for better graphics increase demands on graphics cards. The demands for better control of the games are not probably as big as for better graphics, because even if we can now find more advanced control input devices such as a wii remote or a gamepad, there are still many games and players staying with the traditional mouse and keyboard control. It does not matter, if it is FPS, adventure, role playing games known as RPGs, construction and management simulation games, life simulations, vehicle simulation, strategy or any other genre, the players are used to a very similar game controls as it was 15 years ago:

Mouse movements can change the view point, a movement direction or simply moves the mouse focus. This can be often performed in combination with the keyboard – keys WASD and/or arrow keys. The left mouse button is mapped to the most often used action – to run an active object (to fire a weapon, to select an object, etc.), if there is already a selected object, it can designate an application object, which is similar to drag-and-drop style. The right mouse button is often mapped to the second mode of control, which is not as often used as the one on the left button or it simply provides a context menu. The wheel is often used for scrolling in selected menu or for zooming in and out. The other keys on the keyboard can be mapped to other game functions; which function is

mapped to which key is dependent on how often the function is used and how far the key is from a basic hand position - usual basic hand position is on WASD keys and therefore Q, E, R, F, Shift, Ctrl, Alt and Space keys are often included among the control keys.






4.3.2 Commercial and Free 3D Graphic Editors

There are many different types of 3D editors. Some are designed for technical drawing²⁵, some for 3D animation, and others for art and design. Some, because of their broad functionality, require many hours of training, while other editors can be handled by a beginner. Some are free, others paid. Some are more widespread among users than others. We can find various kinds of controls in these editors. It depends on the type of user (beginner/advanced) an editor is intended for and what is the subject of modeling. For simple comparison we have selected the 5 common editors, covering all the aforementioned categories: Blender (Diz, 2010), Autodesk® 3ds Max® (Autodesk, Inc., 2010), Autodesk® Maya® (Autodesk, Inc., 2010), Google SketchUp (Google, 2011) and Pixologic ZBrush (Pixologic, Inc., 2011).

Blender is an open source editor similar to technical editors, but many of its functionalities are intuitive and thus easily accessible to beginners. 3ds Max and Maya are complex editors with a wide functionality, both for 3D animation (almost on the level of technical type), both proprietary. They differ in the specific features and even in some basic controls, since each of the editors was originally developed by a different company. SketchUp in its basic form is intended primarily for beginners, it is free and it is also one of the technical editors. ZBrush is, unlike the previous, an editor for artists, because the principle of objects is fundamentally different - it uses the metaphor of shaping clay, and is proprietary.

²⁵ For engineers: CAD/CAM, rapid prototyping, ...

Table 5: View and scene manipulation differences in five 3D graphical editors

					
	Blender	3ds Max	Maya	SketchUp	Zbrush
default number of views	1	4	4	1	1
translation the scene	Shift+MMB +MM	MMB+MM (+Shift)	Alt+MMB +MM	Space bar +MM (x&y axes) and MW (z axe)	Alt+LMB +MM*
rotation the scene	MMB+MM	navigation cube in right up corner	ALT+LMB +MM	MMB+MM	LMB+MM*
zooming the scene	MW	MW	ALT+RMB +MM	MW	Alt+LMB-Alt+MM*
changing view	0,1,2,3,4,5,6,7,8,z	click at navigation cube, t, b, f, l, c	--	--	Shift+RMB+MM






MM mouse movement
LMB left mouse button
MW rolling mouse wheel
RMB right mouse button
MMB middle mouse button (pressed wheel)
* if out of the object, otherwise RMB instead of LMB

Table 5 shows the fundamental differences in selected editors for viewing - possible view choices as well as simple change of the scene orientation.

As it can be seen, more technical editors provide 4 views, while one view is more understandable for beginners. Basic functions such as a function to control the view and the scene differ between editors, too. Table 6 shows differences in the use of object from insertion and manipulation, up to deletion. Of special interest is the fact, that every editor has three modes (moving, turning, scaling) between which it is necessary to switch. Only Blender has implemented recognition of gestures and the system itself can automatically detect the mode (it is obviously slower than a keyboard shortcut, but very useful for beginners). These three modes are usually visually clearly distinguishable (see example in Figure 16). In all editors the operation of

inserting a new object requires more interaction steps; the shortest way is to click on the toolbar.

Table 6: Object manipulation differences in five 3D graphical editors

					
	Blender	3ds Max	Maya	SketchUp	Zbrush
insert a new object	press Space bar for context menu and then click Add then click type of object	in right panel click first tab Create and then click type of object	click type of object from toolbar	no object insertion, only extrusion of an existing 2D object (triangle, rectangular, circle) by MM	no object insertion only import and surface modeling
place, orientation and size of inserted object	on cursor position, oriented toward actual view	aligned according to axes	on mouse click position, oriented toward actual view, size respective to MM	--	--
snapping and aligning to existing objects	Shift+S invokes context menu with snapping possibilities	Alt+A+click on a point invokes settings menu for aligning	--	--	--
translation/ rotation/ scaling the object	select the mode on a toolbar or press g/r/s and then LMB+MM; can be used gestures	select the mode on a toolbar or press w/e/Ctrl+e and then LMB+MM	select the mode on a toolbar or press w/e/r and then LMB+MM	select the mode on a toolbar or press m/q/s and then LMB+MM	select the mode on a toolbar or press w/e/r and then LMB+MM
deleting (selected) object	X key	Delete key	Delete key	Delete key	delete button in list of subtools

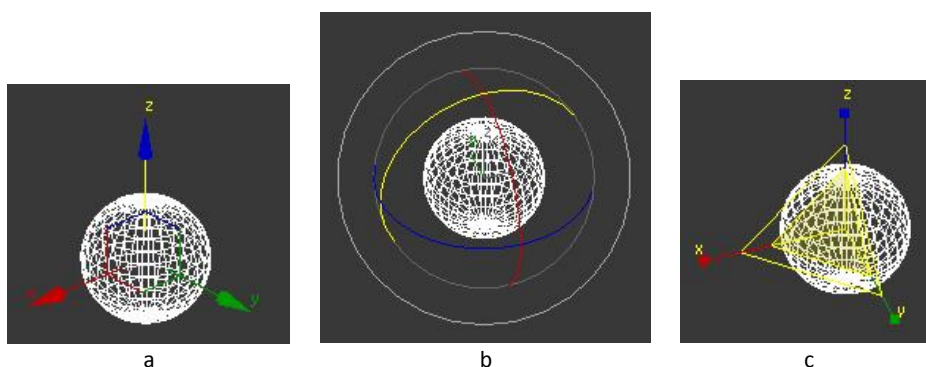


Figure 16: Illustration of different editing modes in 3ds Max editor
dragging (a), rotating (b) and scaling (c)

4.3.3 Educational 3D Editors

There are different educational pieces of software working with 2D or 3D covering various mathematical areas (Math Forum @ Drexel, 1994-2011). Our interest is in editors that simulate task for 3D geometry drawing²⁶ resembling drawing on a paper with a ruler and a pencil. The following list of applications only partially fulfilled our task:

- Geometria (Dumitrascu, 2000-2011) – the user is drawing by setting input from keyboard and not by mouse (but it supports setting problems and then searching for solution or creating the standard one)
- Géoplan-Géospace (AID-CREEM, 2009) – it requires a text input, often mathematical functions
- Geometer's Sketchpad (Jackiw, 2009), Cinderella (Kortenkamp & Richter-Gebert, 2000), Geogebra (International GeoGebra Institute, 2011) and Geometry Master (Caltrox Educational Software, 2011) – they work only with 2D geometry (but they are well-designed)
- GeomSpace (Popa, 1999-2011) and Mathsay (Golden Oct. Computer Network Service Co.,Ltd, 2011) – the user needs to work with digits (a lot of input fields)
- Yenka (Crocodile Clips, 2000-2011) – contains a lot of interactive animations working with 3D geometry, but none working with drawing.

²⁶ We did not pay attention to the editors that are programmable software such as Maple, Mathematica, MatLab or educational software using Logo and/or turtle graphics.

Majority of these solutions belong to a Dynamic Geometry field. “Dynamic Geometry is the theory of construction-like descriptions of function-like objects under parameter changes.” (Kortenkamp U. , 1999) The main focus in this field is on ambiguities, which occur when the user interacts with a construction and moves base points. In our research we do not deal with this problem. We are focused on user-friendly interface.

Archimedes Geo3D and Cabri 3D

Finally the last two applications that meet our requirements are Archimedes Geo3D (Goebel, 2008) and Cabri 3D (de Cotret & de Cotret, 2005), both proprietary. Evaluation of these applications shows following characteristics of their user interfaces:

Both applications have a wide range of drawing tools – contain all the essential possibilities of working with elementary objects, which are well-arranged and grouped in a toolbar (see Figure 17 and Figure 18).

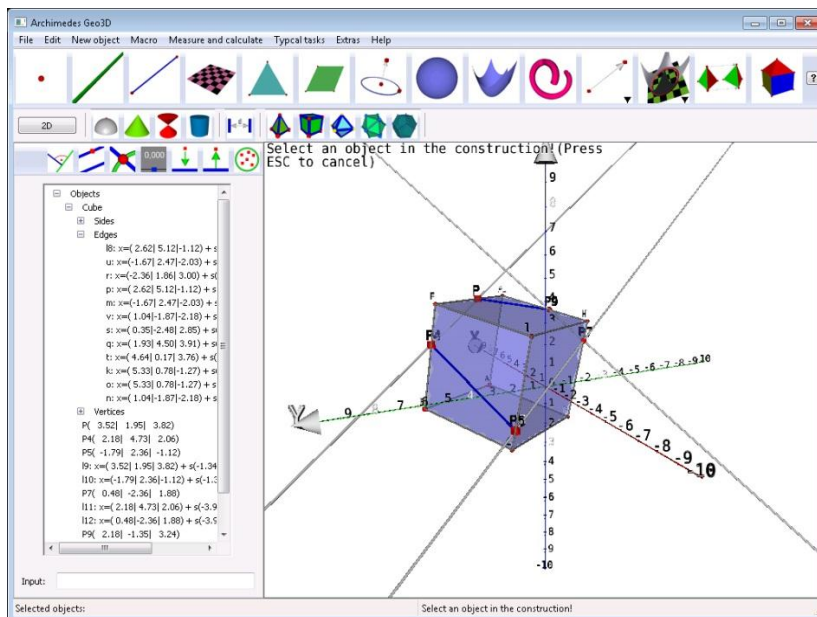


Figure 17: Screenshot of Archimedes Geo3D application

None of these two applications provides shortcuts to work with tools, thus the drawing consists of relentless moving and clicking mouse cursor over the toolbar and back to the scene. Equally arduous is a selection of one of the grouped tools (for example group for adding a line: add a segment, prolonging an existing segment, add a line defined by two points or defined by a point and parallel line, etc.). Some of these

tools can be automatically detected by the system itself, but the majority has to be chosen directly by the user. The next feature of Archimedes Geo3D is a basic axial cross with numeric axes that on one side can assist in the orientation in space, on the other side may be constraining. The base (horizontal) plane in Cabri 3D can cause the same problem. Both applications have a problem with infinite objects (line, plane), because their size/length often make the whole scene unclear.

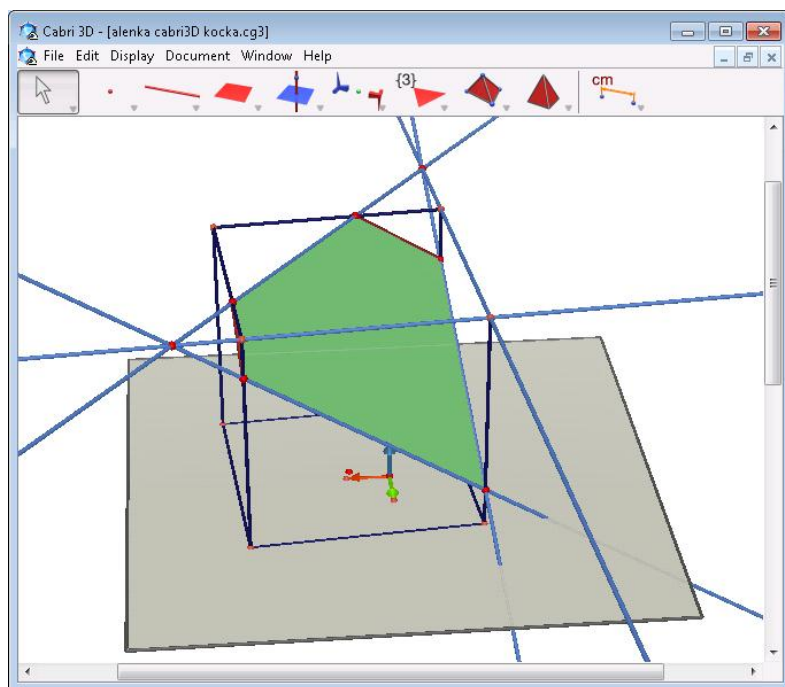


Figure 18: Screenshot of Cabri 3D application

Both applications provide hints during inserting a new object. In Cabri 3D it is implemented in form of a textbox on the mouse position (see Figure 19), the hovered object changes its color and starts to vibrate. According to the mouse position (which object it hovers above/what is possible to do) the content of hint changes – it informs the user what can be done at the current mouse position.

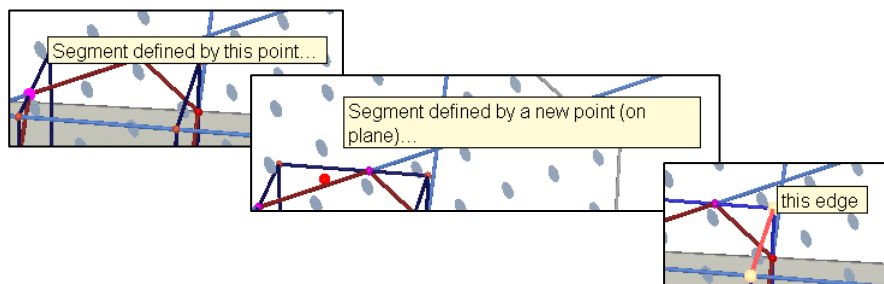


Figure 19: Examples of hints for adding a new segment in Cabri 3D

In Archimedes Geo3D the simple text hint is at upper left corner of canvas, the same hint is at status bar. Next to the mouse position the label of the hovered object is displayed. If the addition of a new object requires more than one click (e.g., a new segment), the system does not show a preview of user's action (e.g., the segment which ends at the mouse position until the second click). Moreover, the selected tool-button is not visually emphasized (thus the user cannot be sure if it is really pressed = missing feedback). Also neither hovered nor selected object in the scene is visually emphasized. These features are significant usability deficiencies of Archimedes Geo3D user interface.

Although Cabri 3D has better user interface, it does not work very well with drag-and-drop interaction style (neither does Archimedes Geo3D). By drag-and-drop the user can zoom or rotate the object, or rotate the scene, or make the dynamic change, e.g., move a point. But it does not work when the user wants to extend or shorten an object. Another negative feature is impossibility to simply interact with overlapped object. The user has to rotate the scene, sometimes even zoom it in (or use another complicated bypass) to get rid of this overlap.

Construct3D

Continuing in the list of relevant existing projects we have to mention also the project Construct3D (Kaufmann & Schmalstieg, 2002) (Kaufmann, Schmalstieg, & Wagner, 2000). We excluded it from the previous list because it works with augmented reality (AR) requiring very specific hardware: head mounted displays, special pen and several tracking cameras (see Figure 20). More recently (Kaufmann, 2009), the author has introduced AR for dynamic differential geometry education in a wide range of ways. For instance, using the AR tool, teachers and students can intuitively explore properties of interesting curves, surfaces, and others.



Figure 20: A student working with Construct3D in our standard AR lab setup with a head mounted display (Kaufmann, 2009)

Hopefully the Construct3D and research connected to it shows the future of educational 3D geometry system. Since its user interface does not use the WIMP style, we do not include more detail about it here.

Planéta vedomostí (The Planet of Knowledge)

The last solution, which is described in this section, is Slovak portal Naucteviac.sk (Agemsoft, 2011). Here are collected the best electronic educational materials from Slovak teachers enriched by professional interactive animations. This portal is part of the Planéta vedomostí project, which Slovak Ministry of Education plans to introduce in Slovak schools (Slovak Ministry of Education , 2011). We focused on the part dedicated to 2D and 3D geometry, especially its interactive animations. Our experience with these animations is summarized in following lines.

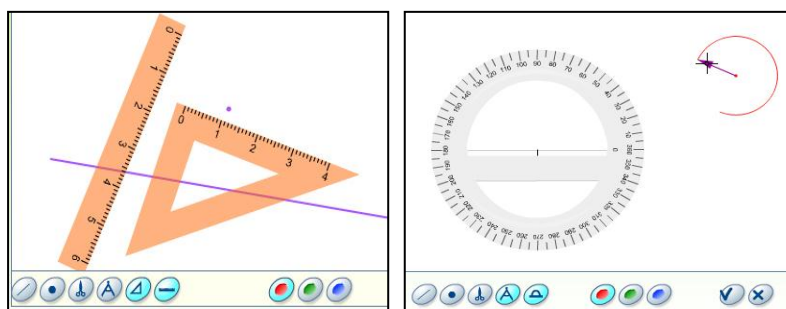


Figure 21: Screenshots of two tasks with interactive 2D scene from Naucteviac.sk portal (Agemsoft, 2011)

Apart from the fact that it is all in Slovak, the simulations of 2D drawing seem very intuitive, because they display the real rulers (the straight one, the triangular one and the protractor), compasses are not so well depicted (see Figure 21). The process of interactive drawing tries

to have traditional properties such as snapping the point to the right position, but it is not very natural: it results in frequent false clicks and there is only one way how to correct it – to delete the object and try to click it again. This feature is very user unfriendly.

This portal contains 3D animations, which can be played and stopped as a video (see left side of Figure 22) or in form of rotating object – the user can rotate it using 6 rotation buttons, no drag-and-drop implemented (see right side of Figure 22). There is no possibility of direct manipulation with 3D objects.

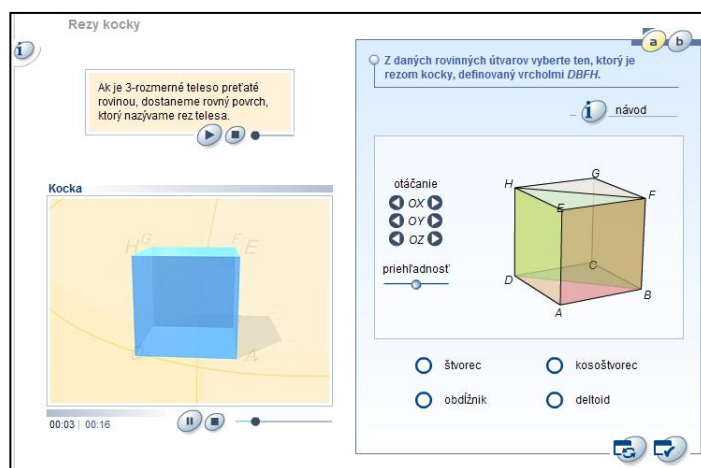


Figure 22: A lesson with 3D animation from Naucteviac.sk portal (Agemsoft, 2011)

4.3.4 Summary

All of the mentioned solutions have its pros and cons. Games typically uses following control elements:

- mouse
- WASD keys
- Shift, Ctrl, Alt and Space (and other) keys.

This makes them easy to control, but they have very narrow range of interaction as regards the creation of new objects.

3D editors have on the other hand wide range of tools for creation of a new and diverse object. But since every editor use different key/mouse control for it, they provide the user with very good feedback by visualizing already selected mode/tool/object in form of

- changed attributes of selected object (color, highlight, vibration),
- changed shape of the selected object envelope (see Figure 16),

- changed shape of mouse pointer,
- hints (in textual form)

and they

- show the preview of object during its creation, when more than one click is needed,
- snap an object to another object or a grid,

Although 3D editors are powerful tools with many useful features, they do not cover requirements for drawing procedures needed in geometric constructions.

The last category is educational 3D geometry editors. Taking into account only the best ones (Cabri 3D and Archimedes Geo3D), one of their disadvantage is (apart from Naucteviac.sk) that they offer full control to the user. On the other hand, the portal Naucteviac.sk does not allow the teachers to create complex interactive tasks and educational lessons for students – the teachers are dependent on developers. The main advantage of this portal is its separation of teachers and students

- students does not have the same rights for editing the scene as teachers,

which is often necessary. Moreover, only this portal provides

- a feedback on the accuracy of the student's solution.

Some of mentioned solution used

- shortcuts,

some not, but without reading manual, it was difficult for novice to use/remember them. The last issue is

- infinite objects, which make the user disoriented in the space.

5. Interaction with Distant Objects

Our analysis has shown (see section 4.1), that there are many different ways and situations when pointing is useful and it is often done by a laser pointer, the dot, which is captured by a camera. This method works only if the object pointed at is in the reasonable proximity, so the computer vision algorithms give satisfactory results – they detect a point within the image (shot) captured by a camera. Moreover, this detected point (of the laser pointer beam) has to be small enough to identify the pointed object. For long distances it does not work. Here we set our ***goal to propose and verify a method of computer aided interaction with remote objects within an outdoor environment using common and affordable equipment (including input and output devices)***. There are different possible and expensive solutions, but because we are focused on affordable devices – our method uses only a computer, a web-camera, a laser pointer and a glass pane.

5.1 The User's Line of Sight Calculation

The core of the problem is when the object pointed at is too far away. In this case the neither direction of the laser beam nor its end (dot on the surface) is detectable/visible in the camera image. At first we explain why this cannot be solved by another type of pointer, e.g. a simple pencil, pen or the index finger: Such a pointer can be tracked by the camera and its end can represent the position the user is pointing at. However, this method creates an unacceptable deviation caused by different positions of the user's hand, the user's eyes and the fixed camera. Even if the camera would be attached to the user's head, which would eliminate problems with position, it would create a problem with

camera calibration – to calibrate the camera for every frame would increase the computing load too much.

To solve this problem, using only affordable devices, we propose to use a web-camera, a laser pointer and a transparent glass pane. The camera has to be placed on a fixed position so it is not needed to calibrate it for every frame, but only once at the beginning. The laser pointer has to be as close to the user's eye as possible and parallel with his/her straight view (see Figure 23).

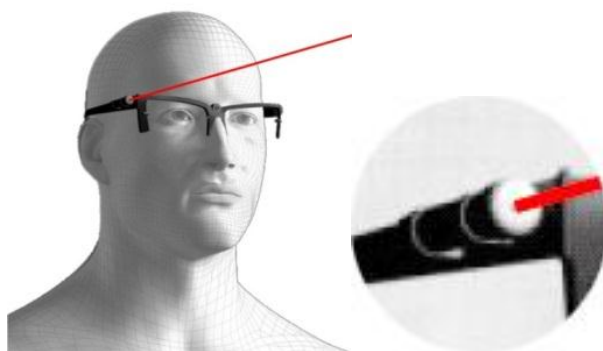


Figure 23: Head mounted laser pointer and detail on laser

This placement of the pointer eliminates the need to calculate the different positions of eyes and the pointer, so there are no deviations caused by their different position. It also enables the user to point the laser beam at the desired target simply by turning his/her head. Finally, the transparent glass pane is used for capturing the laser beam as illustrated in the sketch below (Figure 24):

To calculate the direction of the laser beam, which is an approximation of the user's sight direction, we need to know two different points to define a line, or one point and 2 angles alternatively. The first point for our calculations appears on a glass table, where the light emulated by the pointer falls (Figure 24 - point B). The approximation of the user's sight direction is quite exact due to the fact that the user selects an object simply by pointing at it on the projection plane with the laser beam. We needed to figure out where the second point (the user's eye) is. We cannot take its position as static while everyone moves his/her head frequently and the position of the eye is changing. However, thanks to the positioning of the laser pointer close to the eye, we can just look for this source of light (see Figure 24 - point U).

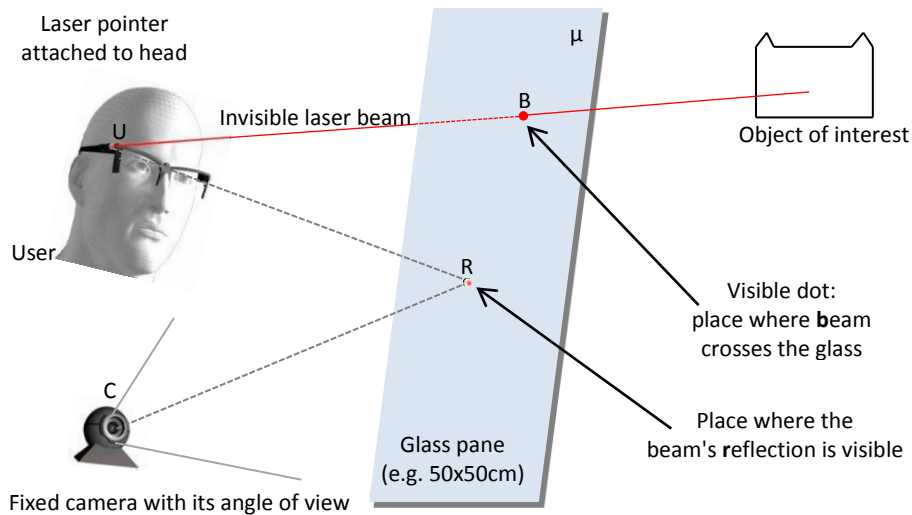


Figure 24: An example user and devices location within user's space and environment

In this case we used a feature of all common laser pointers - they emit not just the one-directional light as a real laser might do. A big portion of the emitted light stays non-directional, so there appears another point on the plane of projection as its natural reflection (see Figure 24 - point R; points R and B are easily distinguishable because R has less red and more white color). Here according the Law of reflection, we are able to calculate the real 3D world spherical coordinates of a light emitter (its azimuth and altitude). Since we are pointing at far objects, the geographical position accuracy within a meter is not necessary and parallel lines provide sufficient accuracy.

Formula derivation for angles

For details of calculation we have outlined a figure (see Figure 25) of the camera with its view representing the captured image.

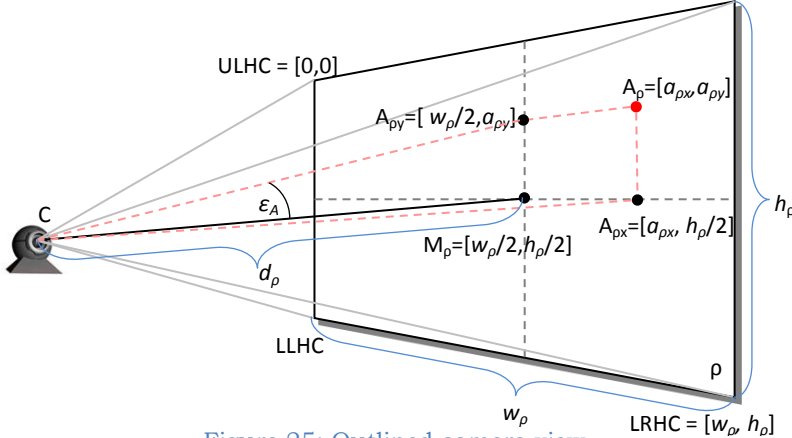


Figure 25: Outlined camera view

The camera (its position is represented by point C) captures an image (represented by a virtual rectangle in a plane ρ in terms of Euclidean geometry). This rectangle has size²⁷ $w_\rho \times h_\rho$ (width \times height), where the upper left hand corner has Cartesian coordinates $ULHC = [0, 0]_\rho$ and lower left hand corner coordinates $LRHC = [w_\rho, h_\rho]_\rho$. The middle of this rectangle is $M_\rho = [w_\rho/2, h_\rho/2]_\rho$. The line $CM_\rho \perp \rho$ and $|CM_\rho| = d_\rho$. Since each camera has in its specification the maximal horizontal (h_{\max}) and vertical (v_{\max}) angle of camera field, we can calculate:

$$\tan\left(\frac{h_{\max}}{2}\right) = \frac{\frac{w_\rho}{2}}{d_\rho} \quad \tan\left(\frac{v_{\max}}{2}\right) = \frac{\frac{h_\rho}{2}}{d_\rho}$$

$$d_\rho = \frac{\frac{w_\rho}{2}}{\tan\left(\frac{h_{\max}}{2}\right)} = \frac{\frac{h_\rho}{2}}{\tan\left(\frac{v_{\max}}{2}\right)} \quad (1)$$

For every point $A_\rho = [a_{\rho x}, a_{\rho y}]$ on the camera image is (using (1)):

$$\tan \alpha_A = \frac{a_{\rho x} - \frac{w_\rho}{2}}{d_\rho} = \frac{a_{\rho x} - \frac{w_\rho}{2}}{\frac{\frac{w_\rho}{2}}{\tan\left(\frac{h_{\max}}{2}\right)}} = \left(\frac{2a_{\rho x}}{w_\rho} - 1\right) \tan\left(\frac{h_{\max}}{2}\right)$$

²⁷ The size of this rectangle is dependent on distance $|CM_\rho| = d_\rho$, which is not set. So we set the distance d_ρ so that the size of rectangle (e.g., in cm) was the same as the resolution of camera image (in pixels). This solution also makes further conversions between the image and rectangle easier. We assume the pixel width is the same as its height.

$$\begin{aligned}\tan \alpha_A &= \left(\frac{2a_{\rho x}}{w_\rho} - 1 \right) \tan \left(\frac{h_{max}}{2} \right) \\ \tan \varepsilon_A &= \left(\frac{2a_{\rho y}}{h_\rho} - 1 \right) \tan \left(\frac{v_{max}}{2} \right),\end{aligned}\tag{2}$$

where α_A is the size of the angle $\angle CA_{\rho x}M_\rho$ and ε_A is the size of the angle $\angle CA_{\rho y}M_\rho$.

Now we need to recalculate all these angles into world spherical coordinates with origin in C – camera (horizontal coordinate system, see Figure 26):

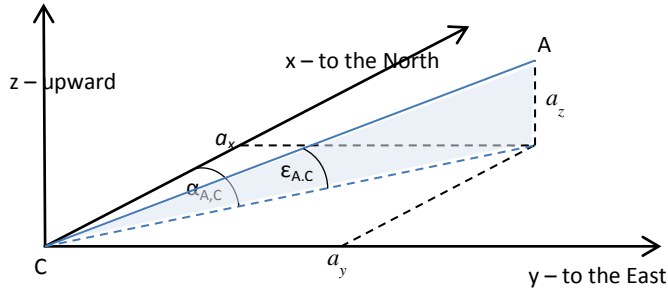


Figure 26: Horizontal coordinate system with origin in C

To do so, we need to know the view angle of the camera – its azimuth²⁸ and altitude²⁹ (α_C and ε_C)³⁰, which gives us the point A in the world coordinates $A = [r_{A,C}, \alpha_{A,C}, \varepsilon_{A,C}]_{C, \text{horizontal}}$, where

$$\alpha_{A,C} = \alpha_A + \alpha_C \text{ and } \varepsilon_{A,C} = \varepsilon_A + \varepsilon_C,$$

These can be transformed using equation (2) in:

$$\begin{aligned}\alpha_{A,C} &= \alpha_A + \alpha_C = \tan^{-1} \left(\left(\frac{2a_{\rho x}}{w_\rho} - 1 \right) \tan \left(\frac{h_{max}}{2} \right) \right) + \alpha_C \\ \varepsilon_{A,C} &= \varepsilon_A + \varepsilon_C = \tan^{-1} \left(\left(\frac{2a_{\rho y}}{h_\rho} - 1 \right) \tan \left(\frac{v_{max}}{2} \right) \right) + \varepsilon_C\end{aligned}\tag{3}$$

Final calculation is a conversion of origin of world horizontal coordinate system from C – camera to U – user (see Figure 24). Here we

²⁸ The **azimuth** of is the angle of the object around the horizon, usually measured from the north point towards the east, in this work we mark it with Greek letter α .

²⁹ The **altitude**, sometimes referred to as **elevation**, is the angle between the object and the observer's plane (local horizon), in this work we mark it with Greek letter ε .

³⁰ Since web-cameras can be turned only in horizontal and vertical direction, we do not need any other correction for camera view angle (such as rotation around its own axis)

need to know the position of the glass pane (plane μ) relative to the ground (plane ν) and the distance h between the glass pane and the user alternatively the camera. The natural position for glass is either vertical or horizontal. The following equations work with a horizontal glass pane³¹ ($\mu \parallel \nu$), but for a vertical glass pane calculations are very similar. Using Cartesian coordinate system with origin in U we have:

$$\begin{aligned}
 |UC| &= d = (d_x h, d_y h, 0)_{U, \text{Cartesian}} \\
 A &= [a_x, a_y, a_z]_{U, \text{Cartesian}} = [r_{A,U}, \alpha_{A,U}, \varepsilon_{A,U}]_{U, \text{horizontal}} = [r_{A,C}, \alpha_{A,C}, \varepsilon_{A,C}]_{C, \text{horizontal}} \\
 a_x &= r_{A,U} \cdot \cos \varepsilon_{A,U} \cdot \cos \alpha_{A,U} = r_{A,C} \cdot \cos \varepsilon_{A,C} \cdot \cos \alpha_{A,C} + d_x h \\
 a_y &= r_{A,U} \cdot \cos \varepsilon_{A,U} \cdot \sin \alpha_{A,U} = r_{A,C} \cdot \cos \varepsilon_{A,C} \cdot \sin \alpha_{A,C} + d_y h \\
 a_z &= r_{A,U} \cdot \sin \varepsilon_{A,U} = r_{A,C} \cdot \sin \varepsilon_{A,C} + 0
 \end{aligned} \tag{4}$$

If $A \in \mu$, then $a_z = h$.

In our captured image, there are two dots (two clusters of pixels): one where the beam crosses the glass (point $B_\rho \in \rho$ which is projection of point $B \in \mu$) and one natural reflection (point $R_\rho \in \rho$ which is projection of point $R \in \mu$). The coordinates of point B_ρ alternatively R_ρ are calculated as cluster's center of gravity using its pixels' coordinates. Using equations (3) we can calculate azimuths and altitudes for points B and R ($\alpha_{B,C}, \varepsilon_{B,C}, \alpha_{R,C}, \varepsilon_{R,C}$):

$$\begin{aligned}
 B &= [r_{B,C}, \alpha_{B,C}, \varepsilon_{B,C}]_{C, \text{horizontal}}, \quad R = [r_{R,C}, \alpha_{R,C}, \varepsilon_{R,C}]_{C, \text{horizontal}} \\
 \alpha_{B,C} &= \tan^{-1} \left(\left(\frac{2b_{\rho x}}{w_\rho} - 1 \right) \tan \left(\frac{h_{\max}}{2} \right) \right) + \alpha_C \\
 \varepsilon_{B,C} &= \tan^{-1} \left(\left(\frac{2b_{\rho y}}{h_\rho} - 1 \right) \tan \left(\frac{v_{\max}}{2} \right) \right) + \varepsilon_C \\
 \alpha_{R,C} &= \tan^{-1} \left(\left(\frac{2r_{\rho x}}{w_\rho} - 1 \right) \tan \left(\frac{h_{\max}}{2} \right) \right) + \alpha_C \\
 \varepsilon_{R,C} &= \tan^{-1} \left(\left(\frac{2r_{\rho y}}{h_\rho} - 1 \right) \tan \left(\frac{v_{\max}}{2} \right) \right) + \varepsilon_C,
 \end{aligned} \tag{5}$$

where known are coordinates of B_ρ and R_ρ from the image captured by camera: $B_\rho = [b_{\rho x}, b_{\rho y}]_\rho$, $R_\rho = [r_{\rho x}, r_{\rho y}]_\rho$, camera resolution $h_\rho \times w_\rho$, camera field with angles h_{\max} , v_{\max} and camera viewpoint α_C, ε_C .

³¹ In this case the distance h is measured as the distance between the ground and the glass pane (its underside); $|\nu C| = |\nu U| = h$

Our unknowns are angles of B for origin U – $\alpha_{B,U}$ and $\varepsilon_{B,U}$ To calculate them, we used the fact that R is a reflection and equations (4):

$$\begin{aligned} B &= [b_x, b_y, b_z]_{U, \text{Cartesian}} = [r_{B,U}, \alpha_{B,U}, \varepsilon_{B,U}]_{U, \text{horizontal}} = [r_{B,C}, \alpha_{B,C}, \varepsilon_{B,C}]_{C, \text{horizontal}} \\ b_x &= r_{B,U} \cdot \cos \varepsilon_{B,U} \cdot \cos \alpha_{B,U} = r_{B,C} \cdot \cos \varepsilon_{B,C} \cdot \cos \alpha_{B,C} + d_x h \\ b_y &= r_{B,U} \cdot \cos \varepsilon_{B,U} \cdot \sin \alpha_{B,U} = r_{B,C} \cdot \cos \varepsilon_{B,C} \cdot \sin \alpha_{B,C} + d_y h \\ h &= b_z = r_{B,U} \cdot \sin \varepsilon_{B,U} = r_{B,C} \cdot \sin \varepsilon_{B,C} + 0 \end{aligned} \quad (6)$$

$$\Rightarrow \quad r_{B,U} = h / \sin \varepsilon_{B,U} \quad \text{and} \quad r_{B,C} = h / \sin \varepsilon_{B,C} \quad (7)$$

$$\begin{aligned} R &= [r_x, r_y, r_z]_{U, \text{Cartesian}} = [r_{R,U}, \alpha_{R,U}, \varepsilon_{R,U}]_{U, \text{horizontal}} = [r_{R,C}, \alpha_{R,C}, \varepsilon_{R,C}]_{C, \text{horizontal}} \\ \varepsilon_{R,U} &= \varepsilon_{R,C} \quad (\text{this is characteristics of reflection}) \\ \alpha_{R,U} &= \alpha_{R,C} + \pi \quad (\text{this is characteristics of reflection}) \\ r_x &= r_{R,U} \cdot \cos \varepsilon_{R,C} \cdot \cos(\alpha_{R,C} + \pi) = r_{R,C} \cdot \cos \varepsilon_{R,C} \cdot \cos \alpha_{R,C} + d_x h \\ r_y &= r_{R,U} \cdot \cos \varepsilon_{R,C} \cdot \sin(\alpha_{R,C} + \pi) = r_{R,C} \cdot \cos \varepsilon_{R,C} \cdot \sin \alpha_{R,C} + d_y h \\ h &= r_z = r_{R,U} \cdot \sin \varepsilon_{R,C} = r_{R,C} \cdot \sin \varepsilon_{R,C} + 0 \end{aligned}$$

$$\Rightarrow \quad r_{R,U} = r_{R,C} = h / \sin \varepsilon_{R,C}$$

$$\begin{aligned} r_x &= h / \sin \varepsilon_{R,C} \cdot \cos \varepsilon_{R,C} \cdot -\cos \alpha_{R,C} = h / \sin \varepsilon_{R,C} \cdot \cos \varepsilon_{R,C} \cdot \cos \alpha_{R,C} + d_x h \\ r_x &= h / \sin \varepsilon_{R,C} \cdot \cos \varepsilon_{R,C} \cdot -\sin \alpha_{R,C} = h / \sin \varepsilon_{R,C} \cdot \cos \varepsilon_{R,C} \cdot \sin \alpha_{R,C} + d_y h \end{aligned}$$

$$\Rightarrow \quad d_x = 2 \cot \varepsilon_{R,C} \cdot \cos \alpha_{R,C} \quad \text{and} \quad d_y = 2 \cot \varepsilon_{R,C} \cdot \sin \alpha_{R,C} \quad (8)$$

Substituting evaluated terms of d_x , d_y (8), $r_{B,U}$ and $r_{B,C}$ (7) in expression for b_x and b_y (6) we have:

$$\begin{aligned} \cot \varepsilon_{B,U} \cdot \cos \alpha_{B,U} &= \cot \varepsilon_{B,C} \cdot \cos \alpha_{B,C} + 2 \cot \varepsilon_{R,C} \cdot \cos \alpha_{R,C} \\ \cot \varepsilon_{B,U} \cdot \sin \alpha_{B,U} &= \cot \varepsilon_{B,C} \cdot \sin \alpha_{B,C} + 2 \cot \varepsilon_{R,C} \cdot \sin \alpha_{R,C} \end{aligned}$$

Dividing last two equations:

$$\begin{aligned} \frac{\cot \varepsilon_{B,U} \cdot \sin \alpha_{B,U}}{\cot \varepsilon_{B,U} \cdot \cos \alpha_{B,U}} &= \frac{\cot \varepsilon_{B,C} \cdot \sin \alpha_{B,C} + 2 \cot \varepsilon_{R,C} \cdot \sin \alpha_{R,C}}{\cot \varepsilon_{B,C} \cdot \cos \alpha_{B,C} + 2 \cot \varepsilon_{R,C} \cdot \cos \alpha_{R,C}} \\ \Rightarrow \quad \alpha_{B,U} &= \tan^{-1} \frac{\cot \varepsilon_{B,C} \cdot \sin \alpha_{B,C} + 2 \cot \varepsilon_{R,C} \cdot \sin \alpha_{R,C}}{\cot \varepsilon_{B,C} \cdot \cos \alpha_{B,C} + 2 \cot \varepsilon_{R,C} \cdot \cos \alpha_{R,C}} \end{aligned}$$

and

$$\varepsilon_{B,U} = \tan^{-1} \frac{\sin \alpha_{B,U}}{\cot \varepsilon_{B,C} \cdot \sin \alpha_{B,C} + 2 \cot \varepsilon_{R,C} \cdot \sin \alpha_{R,C}}$$

The last step is to substitute $\varepsilon_{B,C}, \alpha_{B,C}, \varepsilon_{R,C}, \alpha_{R,C}$ according to equations (5), which gives us the final formula for $\alpha_{B,U}$ and $\varepsilon_{B,U}$ (final formulas are not stated, because they are too complicated and gives no added value to this text). These two angles ($\alpha_{B,U}$ and $\varepsilon_{B,U}$) represent the user's viewpoint.

The user needs to give the system only these values:

- Camera's view direction in world coordinates (its azimuth and altitude); it can be easily calculated, when the user sets the camera under the corner of the glass pane aiming camera to the center of the pane, after such setting up the user can give the system dimensions of the pane and its distance from the ground
- Camera's field of view; user chooses it from drop-down list according the camera type
- Observation position coordinates (usually GPS coordinates)

With the observer's position (from GPS) the object of interest lying in his/her line of sight can be calculated. The coordinate system depends on the application domain. This method can work also in the dark or simply when the camera cannot "recognize" the surroundings. It is not necessary for the system to "see" the object of interest – it is sufficient when it is possible to calculate it from a model. The way of acquiring the model of the surroundings is domain dependent. One possibility is to take a picture (if 2D is enough) of these surroundings during good light conditions. Another possibility is, for example, if there is something moving in a predictable way, then its position can be calculated – usually in GPS coordinates.

If the camera cannot "recognize" the surroundings (e.g., in the night or in the dark, foggy weather), the system cannot autocalibrate itself (calculate camera's azimuth and altitude), so the user has to set it manually. This input is critical for the accuracy of further calculations, otherwise the system will provide wrong outputs.

5.2 Interaction with Computer without Mouse and Keyboard

Aforementioned interaction method has a drawback, because it is not comfortable for the user to control the computer in the usual way –

using mouse and keyboard. For this reason, we were looking for another way of interaction. There are several possibilities. We preferred devices that can control cursor movement on the screen (pointing devices). In previous analysis (see section 3.3.1 Other Input and Output Devices) we introduced several types of them, but those more common and affordable mouse-types i.e., require a pad. And the others are more expensive (such as an air mouse³²). We also considered using a remote controller. Laptop remote control is not commonly used. Other option is a common remote TV controller, but it communicates via different protocol³³ (Insam, 2002). Cell phones, using Bluetooth, Wi-Fi or IrDA Data protocol³⁴, can serve as a remote controller as well. Finally we decided to use a laser pointer and camera, because they reduce the demands on the number of different interactive hardware devices.

The camera itself is not sufficient, because

- pointing by a finger or another object requires good lighting conditions,
- the user has to be close enough to the computer and
- the pointing accuracy is not satisfactory, which fundamentally influences the cursor position accuracy on the screen.

To substitute mouse “clicking” we propose to use sound signals (spoken commands).

5.2.1 Laser Pointer as a Mouse

To make it work, the camera has to capture the computer screen. It can be an ordinary web-camera with basic resolution 640×480. For locating the laser pointer dot, computer vision algorithms (openCV library³⁵) are used. The process of retrieving the laser dot position on the screen (and finally to move a mouse cursor to that place) consists of the following steps:

1. Auto-calibrate camera – the system needs to find the position of the display within an image from the web-camera (and calibrate colors) on start-up
2. Capture images with reasonable frequency
3. Using computer vision algorithms to detect a laser dot – a small cluster of pixels colored similar to the laser color

³² <http://www.logitech.com/en-gb/mice-pointers/mice/devices/3443>

³³ IrDA Data for personal computers, laptops, cell phones, PDAs and incompatible IrDA Control for mice, keyboards, remote controls

³⁴ <http://anyremote.sourceforge.net/>

³⁵ <http://opencv.willowgarage.com/wiki/>

(mostly red color) among pixels which are different in two consecutive frames

4. Move mouse cursor to the position within the computer screen, which was determined by the laser pointer

To auto-calibrate the camera, we use a simple technique: we change the color of the entire screen at the start-up (see Figure 27). This creates a significant change in camera image and thus the large quadrangular changed area is detected (when comparing the earlier image from the camera to the current one). For such detection we used a distance function. This function determines the 4 edges: they are found as positions of pixels having the maximum distance from the center of the area (from all the pixels on its border) where 3 out of 4 pixels are not on the same line.



Figure 27: Auto-calibration screen with four different colors in rectangular areas serving for detection of screen angle within camera image

Having determined the position of the screen, the system excludes it from comparing the foreground to the background image since it is used to detect the user's line of sight (see section 5.1). To find a point from the laser pointer on the screen, our algorithm looks for the pixels whose color is similar to that of the laser. It works very precisely, since our GUI does not contain pixels of such color (it adapts according to the laser color). The relative position of the mouse cursor determined by the position of the laser point in the camera image is calculated as follows in equation (9):

$$\text{cursor coordinates} = \left[\frac{c}{c+d}, \frac{a}{a+b} \right] \quad (9)$$

where inputs a , b , c , d are distances of the found point from the top, bottom, left and right borders of quadrilateral, given in coordinates of image from camera.

5.2.2 Voice Commands

For selection and usage of more advanced functions we propose to use voice commands. This requires a speech recognition module. One of the possibility is to develop own module, e.g., speech recognition system based on analyzing a captured voice using Fast Fourier Transform algorithm (Brigham, 1988). This gives the advantage of higher adaptability of such system – every user can define his/her own set of commands to control application.

Another possibility is to use voice recognition as there are already well designed engines even within operating systems (e.g., in Microsoft Windows Vista or even in web-browser Opera). So when it is used such engine, it only remains to come up with appropriate voice commands. One can say the short and easy to remember commands are the best, but such effort can lead to high error rate – recognition engine has higher success rate with longer commands. For this reason the first of the important steps is to choose such commands (words or phrases) which are easily distinguishable by the voice recognition engine even if the user does not have exact pronunciation. The second step is to consider the number of commands. It has to be low enough or they have to be intuitive, so the user can easily remember all the basic ones even when using it for the first time.

Some implementation details and lists of voice commands used in our pilot project can be found in Appendix B Voice commands.

5.3 Object of Interest Identification

To be able to evaluate our method, we needed to choose the application domain and implement our method for it. The important part in this is to implement the calculation for the object of interest. This calculation method is domain dependent. In the following paragraphs the domain is described as well as the principle of further calculations.

5.3.1 Application Domain

For evaluation of our method we chose an outdoor night sky observation where a computer provides visual, textual and spoken information about celestial objects observed and pointed at by the user.

User description

Our target group are people who like to observe stars and want to know their names, which constellation they belong to and so on. Generally, they do not know anything about stars, but the system can

be used also by experienced amateur astronomers. Users can be of any age, but this type of interest is usual for teenage people.

User minimal requirements on the system

To provide information about celestial objects, it has to know where the user is looking. Since the user does not know the object, the computer has to calculate it, at best, completely without burdening the user. Moreover, since the user lies on the ground looking upwards, he/she has to be able to control the computer without a mouse and a keyboard.



Figure 28: Outdoor experimental interaction prototyping – a user is lying under a glass pane placed on cardboard boxes aiming a laser beam to a star

Environment description

The most usual way for observing the night sky is to go out during a clear night, lie down on the ground and look upwards. The darker the night and surrounding is, the better are observation conditions. During our observation the user lies down under a horizontal glass plate (see Figure 28) and uses our pilot project running on his/her computer.

5.3.2 Finding Sky Objects

The pilot project we implemented is named icPoint (read I see point). It complies with our proposed method for determining the direction of the user's line of sight determined by two angles. They serve as input parameters for further calculation – to determine the part of the night sky the user is looking at and according to this to mark the closest visible sky object (see Figure 29 on page 74 or Figure 30 on page 76).

To find an object of interest we needed calculations based on conversions between coordinate systems and data from object catalogues. To accomplish this, we used three basic coordinate systems to designate the position of an object on a celestial sphere. Each of them uses a different fundamental plane and two coordinates for position:

- The horizontal coordinate system is based on the local horizon. It uses azimuth and altitude and depends on the observer's position on the Earth and the time.
- The equatorial coordinate system is based on Earth's equator, or better, to its projection to the celestial sphere called the celestial equator.
 - The first coordinate used in this system is Right ascension of the object; it is the angle between the object and the vernal equinox point (it is the point where the sun crosses the celestial equator on the March equinox) around the celestial equator.
 - The second coordinate is declination, which is the height of the object above the celestial equator.
 - Coordinates in this system depend very little on the observer's time or position on the Earth's surface (there are effects of nutation and precession). Therefore this coordinate system is used for the position of stars and other non-Solar system objects.
- The third coordinate system uses ecliptic as its fundamental plane. Ecliptic is a projection of the Earth's orbit around the Sun to the celestial sphere.
 - The first coordinate is ecliptic longitude, measured around ecliptic from the vernal equinox to the object.
 - The second coordinate is ecliptic latitude, which is the height of the object above the ecliptic.
 - This coordinate system is used for objects in the Solar system – planets, moons, etc.

More information about coordinate systems and conversions between them can be found in literature (Hajduk, et al., 1987) (Pokorný, 1998), formulas are not stated in this work.

To enrich our pilot project by planets, we needed equations for calculating the position of visible planets. These equations can be found in *Astronomical Algorithms for Calculators* (Pokorný, 1998). To achieve the best performance, the positions of only five visible planets and Earth's Moon are calculated (Mercury, Venus, Mars, Jupiter, and

Saturn). icPoint calculates their coordinates in the ecliptic coordinate system and then converts these to the equatorial coordinate system.

When our system identifies a star, it has to convert horizontal coordinates originated from a processed camera image and search in multiple catalogues for visible objects in the specified area of the sky. icPoint is able to create catalogue instances; each catalogue implementation is capable of searching for visible objects near specified coordinates in the equatorial coordinate system.

5.3.3 Information about Sky Objects

As was already mentioned, the icPoint project is capable of identifying sky objects by determining, which object is pointed at by the user. Based on the selected sky object, the system is able to produce a simulated image of the corresponding part of the night sky. It also provides additional multimedia content (see Figure 29), stored in a local database to provide further data expansion.

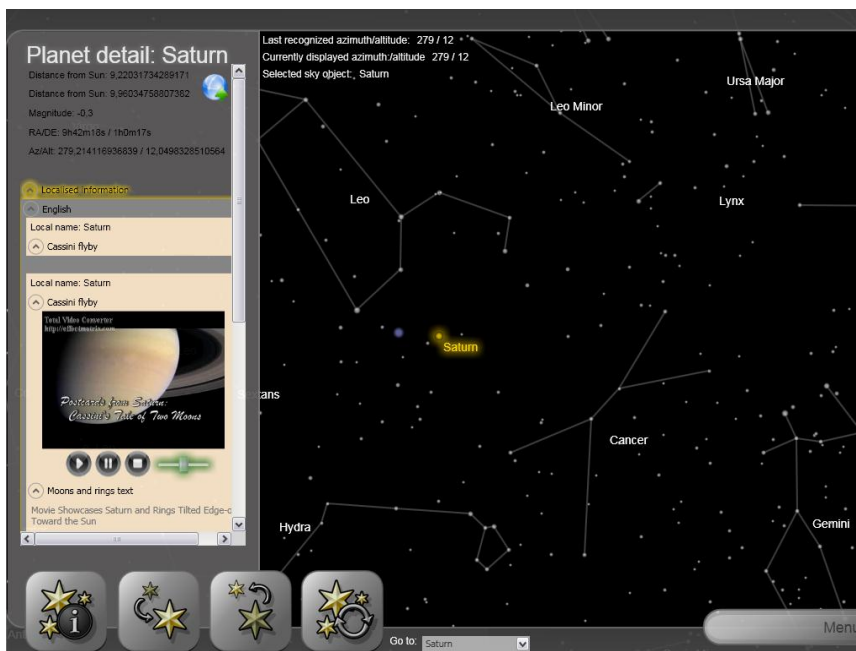


Figure 29: Screenshot of icPoint project with main screen and left side panel, which contains multimedia information about a selected sky object

There are various types of possible multimedia content, for example scientific information about stellar objects, planetary 3D images, but also historical background information (e.g., the discovery of a planet, the history of its name, ancient tales). An interesting idea

might be the possibility to collect stories and tales about stellar objects coming from various cultures of the world. The system enables everyone to add text and multimedia content bound to a sky object sharing it with other users of our system. For this purpose, there was created a collaborative encyclopedia situated on the Internet using Wiki collaborative software (Leuf & Cunningham, 2001). Here all members of the icPoint community can collect information that they found interesting and submit it to the icPoint wiki. The submitted content is not limited to the types listed above, but can also contain videos from space probes, images and videos from telescopes. This wide range of possibilities is accessible via the icPoint main application, but also using a web interface called the icPointWiki web portal.

5.4 Evaluation

Our goal was to propose and verify a new method of computer aided interaction with remote objects within an outdoor environment using common and affordable equipment (including input and output devices). To verify this we set a hypothesis:

Our remote objects pointing method is applicable – if the user points at the object of interest by a laser pointer through a glass pane captured by a web-camera, then the system is able to calculate and identify this object.

To evaluate this, we instructed testers (people from the target group) about the usage of the system and then they got a simple task. This test allowed us to verify three things:

- Proof of Concept - whether the system can correctly identify the user's direction.
- Usability Evaluation - whether the user is able to comfortably use our system.
- Acceptance Test - whether this one use-case meets the requirements of the given specification.

We tested also the other features of icPoint, but for purposes of this work only the listed ones are described in detail.

5.4.1 System Description

The main screen of icPoint consists of a selected part of the night sky. When working with an object (usually a star), it is marked and usually located in the middle of the screen, whether it was designated by the laser pointer on the sky or searched through a find menu placed in the bottom area (see Figure 29 and Figure 30). Four big buttons

dominate the bottom part of this main screen, providing the basic icPoint functionality. They have to be large so the user can easily hit them by the laser pointer dot. On the left side of the screen there is an auto-hiding panel with multimedia information (Figure 29) and on the right side there is an auto-hiding panel with application settings (Figure 30).

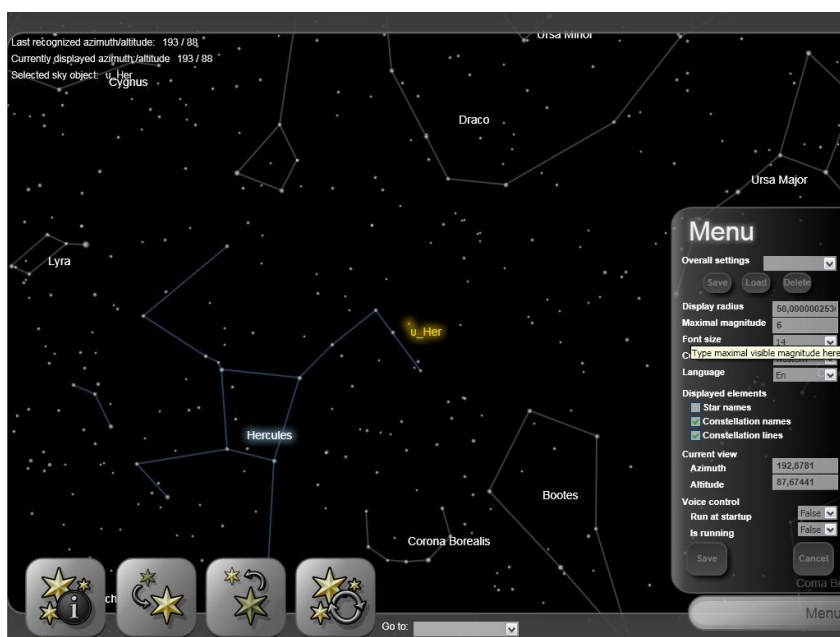


Figure 30: Screenshot of icPoint project with the main screen displaying a part of the night sky with the selected star in the middle; four large buttons placed at the bottom allow controlling icPoint by a laser pointer; panel on the right side contains application settings

Minimal hardware icPoint requirements

- Portable computer (notebook, laptop)³⁶
- USB camera or IP camera with minimum resolution 640x480. The better cam is the one with longer exposure (the camera field of view or its type has to be known)
- Laser pointer fixed in frame in headset (in frame of glasses next to right eye)
- Transparent material (glass) and “legs” for its stabilization (conference table with glass surface is easily available at home)

³⁶ For minimal software requirements check Appendix B Minimal software requirements

- Compass, spirit-level, ruler and known GPS position increase the accuracy

An example of hardware usage is visible on Figure 31 and Figure 32:

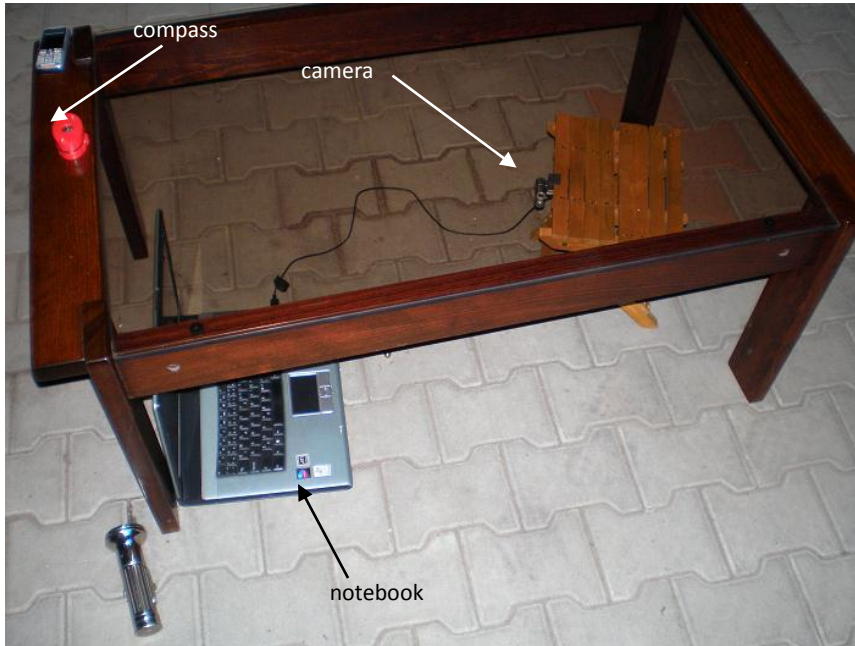


Figure 31: Hardware usage for icPoint – comfortable usage of table with transparent glass

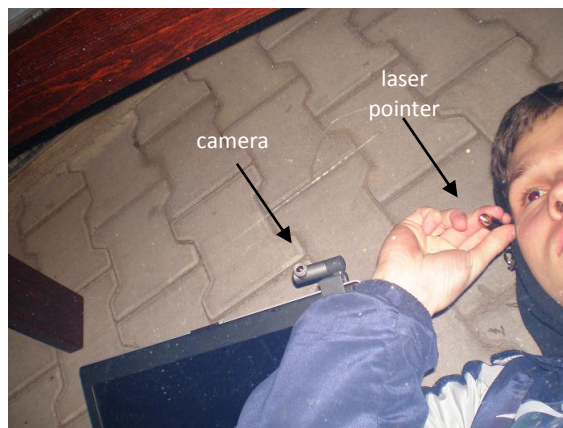


Figure 32: Outdoor interaction with remote objects – a user is lying under a glass table and aiming a laser beam at a star

5.4.2 Experiment

We tested our pilot project during the whole process of its development. This helped us to determine deficiencies in the actual control or computational algorithms and then fix them. The final test involves our development team members and also external testers (altogether 10 testers). The test took place on May 7th 2007. We tested several different features of icPoint. We list here only the one of the user's tasks that verifies sky object detection and identification. It was performed with several stars of the Ursa Major and Bootes constellation.

Task: Aim the laser pointer at the glass in such manner, that the laser dot on the glass designates the chosen star. Then confirm your choice by voice command.

This includes the system's correct determination of the user's angle of view, correct object of interest determination and correct voice command recognition. The user can also ask for information about the object. All of this should be feasible under conditions where the user interacts with the system using only the laser pointer and voice commands.

During this experiment we used scientific data from public astronomical catalogues (Strasbourg astronomical Data Center, 2007). Currently our system uses the Hipparcos catalogue which is based on data gathered by the Hipparcos satellite during the years 1989 to 1993. The same catalogue is used by Stellarium software (Gates, 2009). The catalogue is filtered to contain only visible stars³⁷ (under +6 magnitudes) to speed up searching for stars in the designated area of the sky.

Test Results: The detection of stars was successful – the system correctly detected the area of observed night sky within the system accuracy, which moves around 3 degrees (in means of angular distance on the surface of a sphere). The correct determination of celestial object depended on its magnitude, since our system preferred the brightest one from the closest neighborhood of determined direction. This was the way, how to deal with 3 degrees system accuracy knowing that the users mostly pointed at the brightest stars/objects. This led to perfect (100 % correct) detection of the brightest objects (which were the most frequent case³⁸) and the success rate decreases with increasing magnitude to 0 % (this was the least common case). If the object was detected incorrectly

³⁷ Very bright objects have negative magnitudes, the faintest were of sixth magnitude ($m = 6$) that the limit of human visual perception (without the aid of a telescope). Stars with magnitude more than 6 are not visible by human eye.

³⁸ Depends on actual visibility (observation conditions) and on the user's character

(because of its higher magnitude = less brightness), the user knew the simple voice command, which jumped to the next brightest object in detected area. After one, two or maximum three jumps the correct star/object was selected.

During our tests we gathered from users the following pros and cons:

Pros

- Our method does not require any special or expensive hardware.
- It is portable (if the glass pane is not too big).
- After initial setting up, it is easy to use it any other time, e.g., if the user would set it up on a roof or in a garden.
- Our method can be used even at the night.
- Interaction is natural, even when interacting with the system controlled by a laser pointer or by voice commands – it requires neither a keyboard nor a mouse.

Cons

- It requires a glass pane or a glass table, which can be uncomfortable to carry to the observation location.
- It is complicated to set up the system at the beginning, it requires many steps.
- There are different circumstances which may complicate an observation, e.g., partially reflective ground surface, which creates more laser dots in camera image or any other light distractions.
- Due to the limited size of the glass and spherical character of the sky, it is not possible to observe the entire sky but only a part.

5.5 Discussion

According to our experimental results, we have shown that our hypothesis is correct. The proposed method of determining the user's line of sight and subsequently the determination of the object of interest works correctly.

We proved that this type of application using our method does not need any expensive hardware or any special devices comparing to special devices with an accelerometer and/or gyroscope. All necessary equipment costs less than 25 € (a web camera 10 €, a laser pointer 3 €, a glass pane 10 €/m², a compass 3 €, spirit level and ruler 1 €, smartphones with accelerometer: iPhone 4 for 650 € or Sony Ericsson Xperia X10 for 380 €)

We can also compare icPoint with another solution, which used less common and less affordable technologies such as an accelerometer. It is easier for the user to use such devices for observing distant objects, but it is also more expensive.

One of the disadvantages is the glass, which can be uncomfortable to carry (depending on the size), on the other hand, if someone installs the glass in the garden or on the roof, it eliminates the problem with carrying it as well as the difficult initial setup.

Our solution is very sensitive to a variety of light phenomena which can sometimes occur, e.g., somebody or something moved into the camera view field (even worse when it emits light). It can result in a wrong identification of the object of interest. This is a natural consequence of using input from the camera. The user has to be aware of this and choose the environment with minimal distractions. We have used the known methods for eliminating these unwanted interferences. The second sensitivity issue is accuracy. By observation we found that the error of determining the object is within 3 degrees, which corresponds to the measurement error when entering the initial data entry during setup and also to the distance of the user's eye from the laser pointer. For ordinary observation it is an acceptable error and the user can simply use voice commands to get to the adjacent object.

The method for pointing at distanced objects can be used in any other situation, e.g., pointing at mountain peaks or building roofs (city panoramas), archeological parks. Here, however, the glass would be positioned vertically and calculations for determining the object of interest would differ as well as the objects' data representation. Any of these implementations could be extended to an examination mode, where the user is asked to show the object and the system would determine whether it was the right direction. We created such a system for pointing at objects within indoor spaces (Kovárová, Meszáros, & Zelman, 2008).

6. Interaction Support Based on User Model

The second goal of this work was to propose and verify a method of interaction supported by the user model (see end of section 4.2.1). This model can be created by different methods. The method we are using is based on observing the user's behavior. The user's actions (his/her choices) are recorded, evaluated and incorporated in the user model. When the user is retrieving web information, this model can be used to reduce the demands on the interaction.

6.1 Reducing the Number of the User's Interaction Steps

The analysis has shown (see section 4.2) that retrieving specific information on the web can be very complicated. Especially, when the user already knows the exact place to look for it, but nonexistence of personalized service forces him/her to repeat the same sequence of actions each time he/she is retrieving it. This is not simple information searching (for example looking for the weather or daily news), but in our case information is spread within the different pages of a web site. The user wants only one of the pages, but to get it a selection is necessary. It cannot be displayed without this selection. Since the selection can differ each time, data cannot be automatically retrieved from the web site. For example, it can be searched for a specific food recipe or the nearest bus departure.

To reduce the user's interaction steps in retrieving this type of information (which requires selections), there is only one option – to make this selection by the system instead of the user. This process requires knowing the user's intention. In our case, the user's intention is

estimated by the system on the basis of the user behavior pattern (created from the user monitoring) (see Figure 33). Our goal was not to discover these patterns, but to experiment with already known general simple ones. We represented them by the user model. This model is used to estimate the user's choices and thus it minimizes the number of his/her interaction steps. Recording the user's choices does not burden the user. It does not even require any initial manual setup. This recording together with subsequent estimation creates an adaptable interface, which displays contextual information in a noninvasive manner.

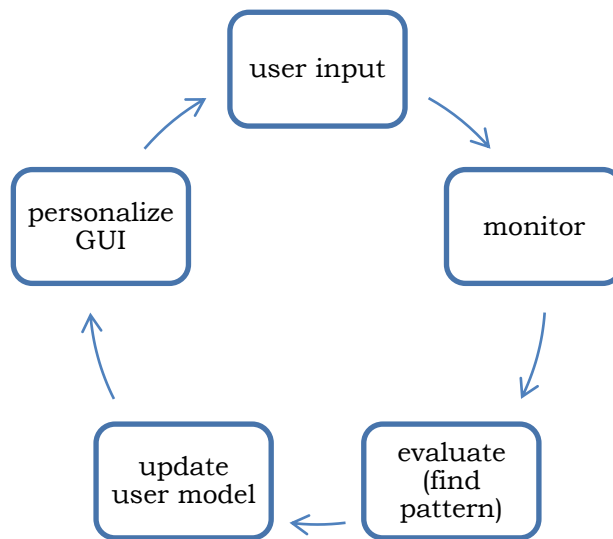


Figure 33: Overview of our personalized interaction approach

The above mentioned patterns are domain dependent, but the most often observed and determined are those that are repetitive and often correlate with time and location. Here repetitiveness represents user preferences, and the time together with the location represents the user's context. We propose to use them in the following way:

- **Repetitiveness** – the system displays the most often used user's choice.
- **User context**
 - **Time** – the system finds the time dependency in repetitiveness, for example when the user is searching for specific information every Monday, the displayed content can be time dependent
 - **Location** – the system finds the location dependency in repetitiveness, for example when the user is often searching

for specific information while connected to the Internet with a certain IP address or at a certain GPS location, the displayed content can be location dependent

Finding the time dependency means to have categories for each hour, each day in a week or in a month, and each month in a year. Location dependency means to group different IP addresses according, e.g., subnet mask: 255.255.255.0 or to group different GPS positions if their distance is less than, e.g., 250 meters.

To reduce the user's interaction steps, the system has to have algorithms, which set *what* and *when* the interface has to display and *how* to solve the situation when the displayed content is not correct (i.e., the user wanted other information = the system estimation was wrong). Using the user's behavior patterns (repetitiveness, whether with time and location dependency or without it), the algorithm follows these rules:

1. *What* to display?

Primary content: the most often used choice (it can be time and location dependent).

2. *When* to display?

Widget can run itself according to the time dependency.

3. *How* to solve incorrect adaptation?

Give choices

- What choices to offer?

Secondary content: the next most often choice(s) (it can be time, location, and previous choices dependent).

6.2 Personalized Information Retrieving for Public Transportation Departures

To add more details about our method, the domain has to be set. The important part in this is to create a model of this domain and then to enrich it by the user model. This gives us the overlay model (Brusilovsky & Millán, 2007). In the following paragraphs the domain itself as well as the domain and user model is described.

6.2.1 Application Domain

For evaluation of our method we chose the domain of public transportation departures. Since a lot of people often use the same (bus)lines, this domain has a simple pattern of the user's behavior. The

most often context is traveling to work (school) and back home. This type of user's behavior can be even very regular.

User description

Our target group is people who use public transportation. They **know** these three facts:

- which line-number they need,
- what the name of the closest (bus)stop is,
- in what direction their route is,

and they **need to know**

- the closest line departures (since they do not remember them)

and they **do not need to know**

- when the line will arrive to their destination
- if line changes are necessary
- which route is the shortest
- which route is the quickest
- which route is the cheapest.

The user uses a computer like device (e.g., smartphone) frequently or with immediate access.

Environment description

It can be any type of environment. We assumed users working with computers at work, or using them at home. It can be only one computer – a notebook. Instead of a computer it can be any type of smartphone. At the start of using our system the Internet connection is required (to download departures). Later, no connection is needed (since data is stored in the database).

The user requirements

The user wants to know several of the closest departure times of a certain line (from a certain stop in a certain direction) with minimal time/effort and minimal manual initial customization. In other words, our system has to fulfill the user's requirements in a way that minimizes the number of interaction steps and accelerates information access.

Domain model

The main element of our domain is a line. This line has a number (1, 2 ...), is of given type (bus, tram ...) and has a route with several stops.

Each stop of the line has its order within the line route, has its time shift – how long it takes from the first stop of the line to the chosen stop, is in a certain direction (towards one or the other terminal stop).

Each stop has a name. The first and the last stop of the route are terminals.

Each terminal has its departure time schedules. They depend on the type of day (regular working day, Saturday, Sunday, public holiday, school vacation, etc.).

All of these relationships and data types are represented in the following domain ontology model (Figure 34):

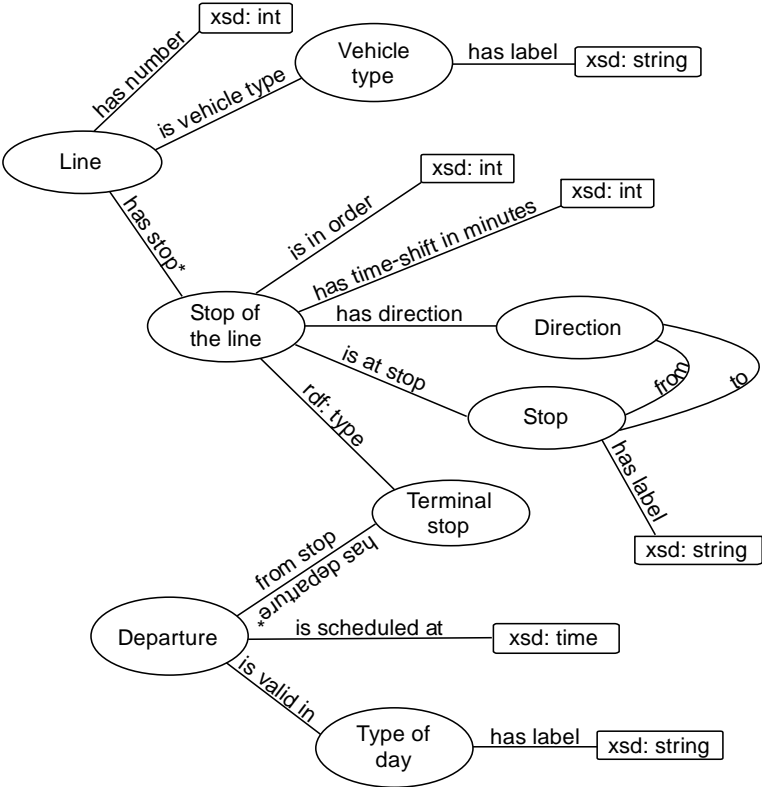


Figure 34: Ontology model of data from public transportation departures

This is the minimal model of domain for our purpose. The model can be extended especially when a different system deals with a broader task, as for example, in Junli Wang and his colleagues work (Wang, Ding, & Jiang, 2005). Their model is not meant for widgets, but it also deals with public transportation. They are oriented to public transport query as a transfer trip scheme, route query and station query. That is a wider range of public transport domain than ours, thus also their ontology model is wider and a little bit different.

6.2.2 Personalization

To provide the user with the closest departure times, the system has to have three inputs (set choices): line number, stop, direction. Without personalization, all of them have to be chosen by the user. Working with the user model, the system can estimate all three inputs.

User model

The user model represents the user's behavior patterns. In our case it is the repetitiveness of his/her choices – the three inputs: number, stop, direction. Since these are already a part of the domain model, the user model is an overlay model of our domain model.

The user's inputs are continuously monitored and since these inputs can be time and location conditioned (contextual), each choice is also categorized:

- ***Repetitiveness***
 - Line number
 - Line direction
 - Line stop
- ***User context***
 - ***Time*** of the request (hour, day, and month)
 - ***Place*** of the request

All of the user's choices are stored in their corresponding category, e.g., in a local database. The displayed content is refreshed either when the system is started or after the change of the context – the time or the location.

The stored categorized data enable the system to adjust the refreshed information to the user by estimating the user's choices (see Figure 35). We choose a subset of data primarily by the user's location, since this context is directly related to the choice of the (bus)stop.

The given flowchart (Figure 35) represents an estimation to the question “*What* to display?” (see page 83). In the case of “*When* to display?” question it is very similar. The system checks whether there is a relevant change first in a location and second in time. The change is relevant if

- it differs from the last one (e.g. the location difference is greater than 250 m and the time difference is greater than 30 min)
- and at the same time there exists a different context related subset of data.

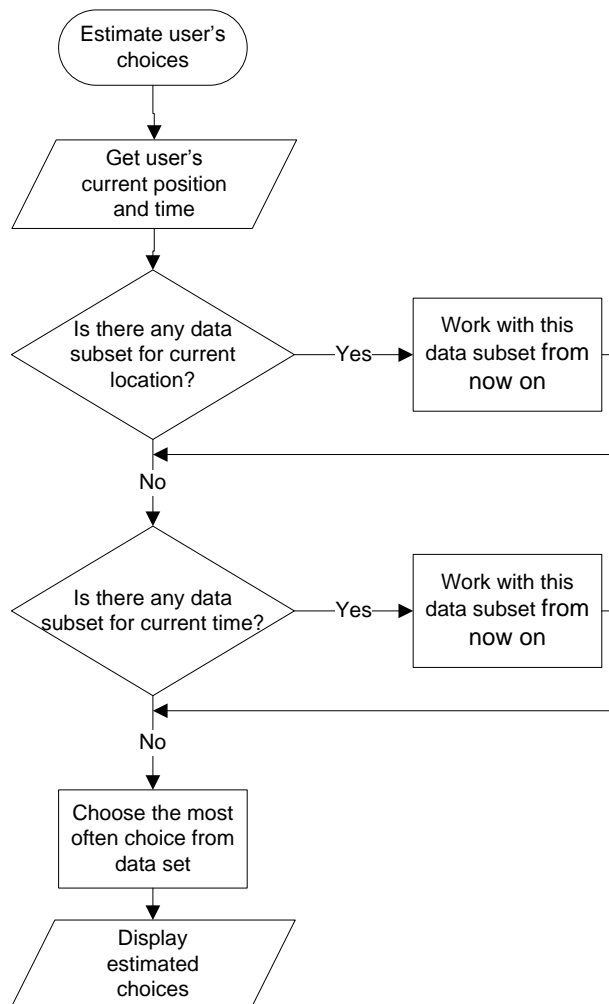


Figure 35: Flowchart for estimation of user's choices

This can either refresh the displayed data to a different one or it can run or turn off automatically the application.

The last question was, “*How to solve an incorrect adaptation?*” If there are no data stored (e.g., the first touch of the user) or if the choice estimation was wrong, the system provides the choices for manual selection (the same one as the choices of any other system providing departures). In the case of the existing subset data for the current location, the lines belonging to this location subset (with corresponding stop and direction) are offered first (e.g., on the top of the drop-down menu or in a quick access bar).

Cold start problem

The above mentioned method has the cold start problem – there is nothing to display when the user model is empty. There are several ways to eliminate it. The two most often ways are:

1. To use a history of other application (e.g., cookies)
2. To let the user set the initial values.

The second one requires manual setting up, but it rapidly increases system effectiveness.

6.3 Evaluation

Our second goal was to propose and verify a method of interaction supported by the user model when the user is retrieving web information requiring selection(s). To verify this we set a hypothesis:

The system adaptation using our user model reduces the demands on the user's interaction and thus accelerates information access and users find this way of information retrieving useful and comfortable.

We used three different methods for three tests:

- Usability Inspection Method – KLM (Keystroke-Level Model, see section 2.1.4) to compare our and other interaction methods.
- Usability Testing Method (see page 11) – Performance Measurement with one task dealing with a cold start done by 10 testers.
- Simple interview dealing with the acceptance, where people from the target group were asked to try to use our application for several days.

6.3.1 System Description

We studied the domain of public transportation departures and found out that there is no service that would monitor the user's requirements and identify patterns in his/her behavior. Usually there are web pages which can remember the user's last choices (cookies) or the user can manually mark a page as a favorite (bookmark). According to our analysis (see section 4.2), when dealing with information retrieving from the web, where the type of information is very specific and short, desktop widgets provide a suitable starting point. Our widget displays in a very little space the closest departures of the chosen/estimated stop, direction, and line of the public city transport (see Figure 36). It uses our proposed method to estimate the user's choices and thus adjusts to the user's needs. In our user model we included only repetitiveness without context (time and position). It

means the most often chosen option is always pre-selected automatically.

As we do not use any other information sources (e.g., browsing history) to find out the user's usual bus stops and bus lines, the widget has an empty local database (except default data) at the beginning.

The displayed information is loaded either from a local database³⁹, or downloaded from the web. When downloading is induced, new data are stored in the local database. Since departure schedules are changed from time to time, these changes need to be translated into the local database update to provide the user with the most up-to-date information (automatically every week, but can be switched off). The system automatically cleans up the database – it erases data that are not used and are old.

The most important system task is to keep fresh data in the displayed area – current departures – therefore they are refreshed every minute.

Widget functionality

The basic widget functionality is to display the upcoming five departures of the selected line from the chosen stop in a set direction. These three choices can be done by the user or estimated by a system. In the case of the user, he/she has to go through three steps, which should be done in proper and intuitive order:

1. Select a line number – from a list within the dropdown-menu (Figure 36, point 1), selection is needed only if the user does not want the one automatically chosen.
2. Change a direction – simple click (Figure 36, point 3), needed only if the widget wrongly proposed the inverse one
3. Choose a stop – from a list within the dropdown-menu (Figure 36, point 2), only those stops are shown that belong to the previously selected line. This selection has to be done only if the automatically chosen stop is not the one wanted. In the case of the first-time line selection, the first stop of the selected line is pre-selected.

³⁹ Since there is no service that would give us the required information on demand (only different web sites) we decided to parse them. Although we chose <http://imhd.sk> as a data source for our widget, since it had a structured html code good enough to parse it to our database, to parse a web page took several seconds due to many irregularities and inconsistencies in it. This was contrary to our goal of time effectiveness. Therefore we needed to store the data in our local database.

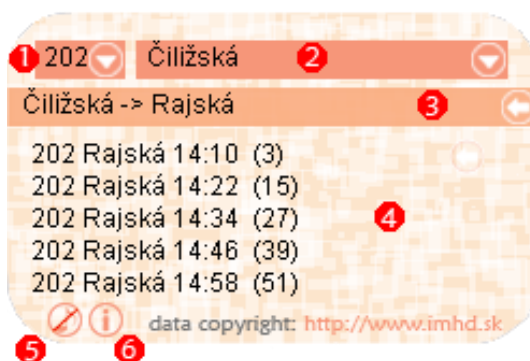


Figure 36: Widget layout description

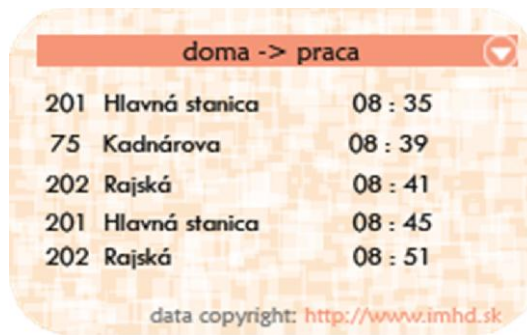
After these three steps, whether they were done automatically or by the user, the upcoming five departures are displayed (from the current time). The widget displays exactly: line number + direction + departure time + time-left in minutes (Figure 36, point 4).

To alleviate the user from continual time checking, i.e., how many minutes remain to a departure, we implemented also one extra feature – sound. The widget can announce the time of the next departure, e.g., "Next bus arrives at 12:00. That is in 3 minutes." Of course, this function can be turned off (Figure 36, point 5).

Finally, every application should have a Help feature (Figure 36, point 6). Our Help contains a user manual.

To make it more user friendly, we gave the user the possibility to set up his/her favorite locations manually. These predefined locations adjust the widget from the first touch and eliminate the cold start effect. The user can choose several lines for every location (with corresponding stops and directions), which he/she usually travels with, for example from home, school or office. The user can name it, e.g., "school->home route."

The output is the same as in the basic functionality, only the upcoming five departures differ in the line number and the name of stop. Departures are ordered in the usual way – according to time of departure (see Figure 37).



doma -> praca		
201	Hlavná stanica	08 : 35
75	Kadnárova	08 : 39
202	Rajská	08 : 41
201	Hlavná stanica	08 : 45
202	Rajská	08 : 51

data copyright: <http://www.imhd.sk>

Figure 37: Widget setup for multiple lines within one route (in Slovak language, translation of route: Home -> Work)

System architecture

Evaluating the pros and cons of different widget APIs, we have decided to implement a widget using the Yahoo Konfabulator (see section 4.2.2). This means we used mainly XML and JavaScript for programming and supported SQLite for our local database. Our system can be divided in the following parts (see Figure 38):

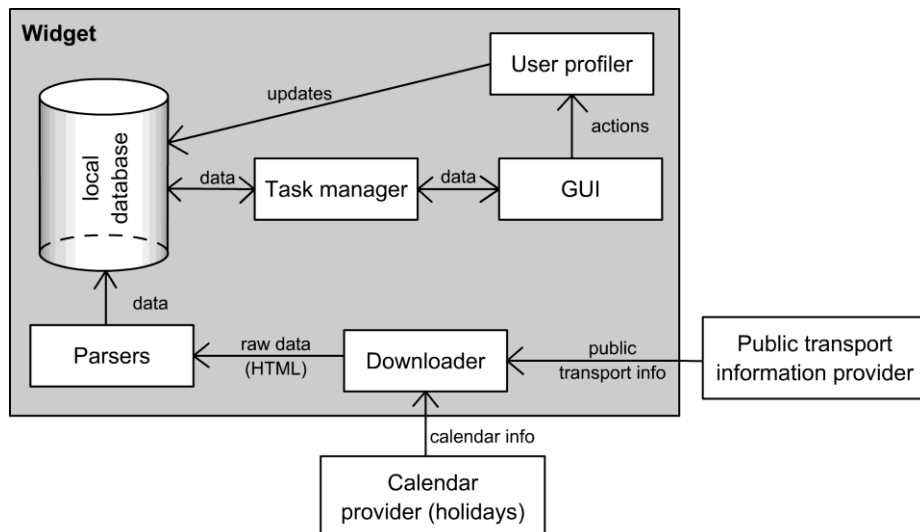


Figure 38: Conceptual architecture
of the public transportation departures widget

GUI – Graphics User Interface, which is used to send data (user choices) to the Task manager and accordingly can ask the Task manager for new data from the local database. The GUI can also send information about the user's choices to the User profiler.

The User profiler updates the number of the user's selections in the database and remembers the user's settings, including his/her favorite locations/routes. Anytime the user chooses a line number, stop or direction, its relevancy raises.

The Task manager

- updates GUI (departures) either because of time or the user's different choice,
- updates the local database (data downloaded from Public transport information provider, if there is an Internet connection) and
- cleans up the local database - due to performance optimization, the Task Manager will erase the least selected lines out of the database after a certain period

The Downloader downloads from the designated web page, therefore an Internet connection is needed when the user wants to download new time tables or a new calendar.

The input of the Parser is raw data (HTML code of a web page), which is parsed and stored in the corresponding columns of the local database – wherefrom it will be loaded for the user as requested.

The physical data model of the widget local database with its description can be found in Appendix B, the section B.3 Widget. For other implementation details see our paper (Kovárová & Szalayová, Semantics in the Field of Widgets: A Case Study in Public Transportation Departure Notifications, 2010).

6.3.2 Experiment

Usability Inspection

To count the number of interaction steps, or more precisely the time required to perform these steps, we used the KLM (Keystroke-Level Model, see end of section 2.1.4). To compare our solution we took three different web sites, which provide the same information in different ways. All of these ways are typical for retrieving the information requiring selections and are very similar to all other solutions dealing with our type of problem. In the following table (Table 7) we summarized all necessary interactions. For a better idea, we calculated it also in seconds using the following times (Kieras, 2001):

- K – Keystroke - Average skilled typist (55 wpm): 0.2 sec.
- T(n) - Type a sequence of n characters on a keyboard ($n \cdot K$ sec.)
- P - Point with mouse to a target on the display (1.1 sec.)
- B - Press or release mouse button (0.1 sec.)
- BB - Click mouse button (0.2 sec.)

- H - Home hands to keyboard or mouse (0.4 sec.)
- M - Mental act of routine thinking or perception (1.2 sec.)

Table 7: The required number of interactions steps for retrieving time of departure

data source	http://www.dpb.sk	http://www.ihmd.sk	http://www.cp.sk
Without bookmark	Type the address + 1 click on a button + ... $T = 0.2*5 + 1.5*3 + 0.2 = 5.7$	Type the address + 1 click on a button + ... $T = 0.2*5 + 0.2 + 1.5 + 0.2 + 1.2*2 = 5.3$	Type the address + ... $T = 0.2*5 + 1.5 + 0.2*2*3 + 0.2 = 3.9$
With bookmarked web site	3 times drop-down list + 1 click on the confirmation button $T = 1.5*3 + 0.2 = 4.7$	1 drop-down list + 1 click on a stop name + $T = 1.5 + 0.2 + 1.2*2 = 4.1$	1 drop-down list + 2 times typing + 1 click on the confirmation button $T = 1.5 + 0.2*2*3 + 0.2 = 2.9$
With bookmarked site using cookies or search field	Does not provide neither cookies nor search field	(search field) type the number of a line + 1 click on a stop name + the user has to check the day and the time $T = 0.2*2 + 0.2 + 1.2*2 = 3.0$	(cookies) 2-3 times drop-down list + 1 click on the confirmation button $T = 1.5*2 + 0.2 = 3.2$
With bookmarked or saved web page (of specified line)	It is no longer available	The user has to check the type of a day and the time, then find him-/herself the closest departure in the timetable $T = 1.2*2 = 2.4$	The user has to check the time and find him-/herself the closest departure in the timetable $T = 1.2$
Feature	Shows timetable for the current day with the highlighted closest departure	It is very simple to bookmark the web page for specified line	Bookmarked web page automatically shows the timetable for the current day

For typing we assume the system (browser / web page) has implemented the autocomplete function so we count 5 keys for typing the address, 3 keys for typing the name of a stop and 2 keys for typing a line number. For a drop-down list we count BB+P+BB ($0.2 + 1.1 + 0.2 = 1.5$). All numbers are times in seconds.

To compare it with our solution, we can use our cold start (or situation when it is necessary to download a new line), wrong estimation and correct estimation times:

- A new line: 1 drop-down list + paste the correct web-page address + ... $T = 1.5 + 1.2 + 0.2 + 0.2 + 1.5*2 + 0.2 = 6.3$ sec.
- When estimation is incorrect: 2 drop-down lists + one click $T = 1.5*2 + 0.2 = 3.2$ sec.
- When estimation is correct: to run application, if it is not $T = 0.2$ sec. (we can also assume, the widget is still running as well as the web browser and thus count $T = 0$ sec.)

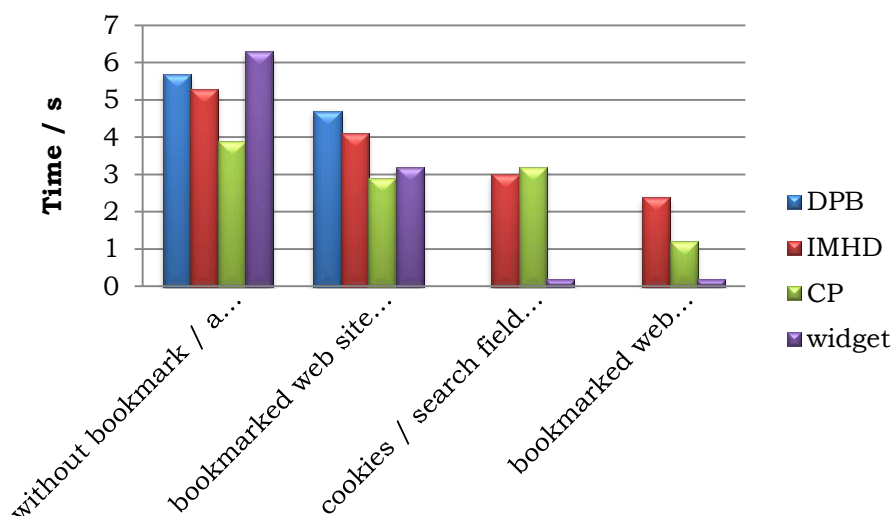


Figure 39: Time consumption comparison for obtaining information from various sources using different ways to speed up search

The graph (Figure 39) shows times for obtaining information on a line departure. Since the majority of users use the same lines, the probability of the correct estimation is high. Thus our personalized widget in most cases reduces the number of interaction steps. It saves the user's time and makes the information retrieving quicker.

User evaluation

We also did two evaluations among the students of the Faculty of Informatics and Information Technologies of the Slovak University of Technology in Bratislava in December 2007.

This test is a simple performance measurement and was performed by 10 volunteers who use the computer on a daily basis.

Task 1: To download the set of a new line (of public transport) to our widget and let it display departures for one of its stops in a certain direction.

Test Results 1: Overall, it took generally less than three minutes for users to find the desired departure.

The long period was caused by a complicated data source – it was not possible to do it automatically and all the steps had to be done by the user. We were aware of this, so we let the widget after its first run display instruction guidelines. But they were usually skipped by the testers. As testers realized during their first attempts that the widget displays only one default line, our guidelines were used to get the information on how to extend the widget's functionality.

The other test dealt with acceptance. We interviewed several of testers (again 10 students of our faculty) after they completed our task.

Task 2: To use our widget for a few days/weeks.

We asked them what they liked and disliked and why.

Result 2: The fact that some of them used our widget also after we finished the testing, shows us that these users were very satisfied, which is what they claimed during our interview. From the interview we gathered the following pros and cons. There are not many of them; the widget was in general rated in a very positive way.

Pros

- One special feature of the widget is sound – the widget can announce the time of the next departure. This feature was also tested (the speech was accomplished by using the Windows functionality of automatic reading of a given text). This voice functionality was evaluated as being very popular by the users.
- Since our testers had not bookmarked web pages for individual lines, the effect of using our widget was for them even more noticeable and appreciated.

Cons

- Testers observed the specific feature of the application - due to data parsing after the URL was set – that the widget got nonresponsive for a moment. (We were aware of this negative feature - it had been documented within the guidelines.)

These pros and cons are related to different implementation details, not to our method.

6.4 Discussion

According to the results of our experiment, we have shown that our hypothesis is correct. The proposed method fulfills our goal. At its best, the user does not have to interact with widgets at all. We assume this is the most often case. The users found it difficult to start to use our widget. This is visible in the result from the second test and also in our graph (Figure 39), where the first violet column is the highest from the group. This undesired result is caused by the information source, whose provider protects data in different ways and thus makes it complicated to use the data in our widget. But after overcoming the initial widget unfriendliness, the users considered the widget as a very pleasant and useful application.

Right now the widget works only for one specific information source, but in principle the final implementation is only dependent on the data provider. The method itself is independent and can be implemented with the same result for any domain dealing with line departures.

An important part of the presented results has been published at international venues endorsed by ACM and IEEE in 2008 and 2010 (see Appendix A section A.1 Publications).

Presently we participate in the project iTransit⁴⁰, which provides the closest departures and even more functionality (route planner, searching according to a stop name, according to a line number, etc.). iTransit already works with the user's context: time and position (see Figure 40), which creates a comfortable interface.



This screen is available after one click, making this solution comparable with ours. We also plan to implement there a user model for repetitiveness and maybe also as a widget for different mobile operating systems.

Our method is unique in that besides using a user model, it also takes advantage of widgets. These two features make information accessible without a single click, while any other solution requires at least one click, often several interaction steps.

⁴⁰ Available at [itms://itunes.apple.com/sk/app/itransit/id389445189?mt=8](https://itunes.apple.com/sk/app/itransit/id389445189?mt=8), more info at www.itransit.sk/



Figure 40: Screenshot of the iTransit application for iPhones: Screen with the closest departures from the closest stops according to actual time and the user's GPS position

Since any widget can be personalized, it can adjust itself to best serve the user, and thus making the retrieving of information more comfortable and quicker. In our case, this adaptation is achieved by monitoring the user's choices and storing the number of selections for each choice in the user model. This method can be applied in any other application domain that deals with regular departures, e.g., logistics or catering. It can be also used in any application domain where repetitiveness is present, e.g., food shopping.

7. Interaction Using Visual Help in Simple 3D Editors

The third goal of this thesis was verification of increasing usability resulting from different methods of visualized information in 3D graphical editors. The main emphasis was on graphical hint for a hovered object, which previews the consequence of object's selection.

The analysis has shown (see section 4.3), that various 3D environments, whether game, editors or educational applications, have different useful features. They help the user to control and use this environment. On the other side, there is still a room for various improvements.

7.1 Improvements of GUI Elements

The GUI design includes elements such as icons, tooltips, help, changes of color or some other attribute, different pointer shape, context menus, menus, etc. To improve some of them, we suggested the following:

- To display keyboard shortcuts directly on tool buttons.
- To create and display “special” keys for most often used tools.
- To show an action preview during scene editing (preview the consequence of the action).
- To reduce the number of buttons in a toolbar.
- To show an infinite object in a finite form with possibility of their extending or shortening.
- To rotate the scene in case of overlapping objects when adding/snapping a new object to an existing (overlapped) one.

Of course, we also suggest keeping already used standards and some habits which users can have from similar applications even if these habits are not standard. The most interesting for us are:

- To display hints in tooltips, an information panel and/or a status bar - short information on the next possible action executable current position of the mouse, especially when the user hovers over an object.
- To allow using the mouse to perform most of operations.
- To change visual feedback attributes of selected/hovered object (color, highlight or vibration).
- To change a shape of the mouse pointer to indicate the type of a mode/action.
- To show the preview of an object during its creation, when more than one click is needed, for example rubber bending.

In the next paragraphs follows the detailed description of our suggestions (see the first list), which were evaluated in our pilot application (see section 7.2):

Shortcuts directly on tool buttons

Using keyboard shortcuts is a well-known method for speeding up the user's work. The problem is that except for shortcuts common for any kind of application, such as ctrl+a, ctrl+s, ctrl+z and so on, each application has its own shortcuts. If a user wants to use these application specific shortcuts, it increases demands on his/her long-term memory. Another aspect is that beginner users usually do not use these shortcuts, because they don't know / remember them. We suggest displaying these shortcuts directly on the tool buttons.

To create and display "special" keys

By "special" keys we mean again keyboard shortcuts, but these consist of only one key and they only switch to the other tool button during the time the key is held (temporary state). It is the same principle when holding Shift key for capital letter(s). We suggest this for the user, who uses two different tools (or more) and needs to switch them very often (and quickly) – in this case a "special" key saves time significantly:

- Original (usual) way: two different shortcuts, which usually consist of 2+2 but at least 1+1 keys
- Quicker way: the user needs to use only one "special" key – hold it for necessary time.

As suggested previously, we propose writing this special key directly on the tool button (as a label), which lowers demands on long-term memory and helps beginners to use it.

Utilizing experiences from games (see subsection 4.3.1), we experimented with Shift, Ctrl, Alt and Space keys.

To show an action preview during scene editing

This idea can be found for example in Microsoft Word 2007, where after hovering over a new format: a preview of reformatted text appears and on rollout the original formatting is restored. This means that the consequence of a selection is visible before the selection itself. We transfer this idea to 3D editor, where a preview of an editing action (the system visualize the consequence of the possible action) will appear on roll over the object and disappear on its rollout. This creates a visual hint for possible user's action (click).

To reduce the number of buttons in toolbar

Working with 3D graphical (educational) editors, there is basic set of functionalities/tools. The more complex is the program, the more buttons (tools) there are. Some of these tools can be grouped in the case when the system can automatically differentiate the correct element of the group. For example such a group can be: adding a parallel line, a perpendicular line, a parallel segment, a perpendicular segment, extending a segment, shortening a segment. A system can recognize the user intention – the chosen element of group – according direction of mouse movement.

To show infinite object in finite form

During our analysis of educational applications working with 3D space (see subsection 4.3.3) we noticed that the infinite objects caused different problems:

- Viewpoint of working area is overfilled by these infinite objects
- The scene looks messy
- The infinite objects can be obstacles when interacting with other objects

In a small survey between users, they suggested to allow making infinite object (especially lines) as long as the user wants – to set his/her own “clipping area”.

As in previous case with infinite objects, the users prefer limited objects to be displayed clipped to the desired size – (especially extendable or shortenable segments)

To rotate the scene in case of overlapping objects

One of the problems of 3D space is overlapping objects. At first, the user does not have the clear idea about the depth. Then the user may try to reach the object, which is in greater depth. Different applications solve it in different ways (e.g., using Tab key to switch to the next object), but most of them allow the user to get the object to the front position. We suggested and experiment with automatic scene rotation when the application detects the user has a problem, e.g. he/she tries to add a point on “intersection” of two skew lines.

7.1.1 Application Domain

For evaluation of our approach we chose an educational 3D editor working with solid geometry. This can help students with lower spatial ability (Cohen & Hegarty, 2007) to utilize it. This is the area, where users work in virtual 3D space with abstract 3D objects. In our case, this 3D space is projected on 2D screen. Since the same principles are involved in all drawing tasks, in order to prove our concepts, we chose one specific task: a cross section of a cube.

Students' task description

There is a cube (only a wire frame model) with 3 different points marked, usually on its edges (Figure 41).

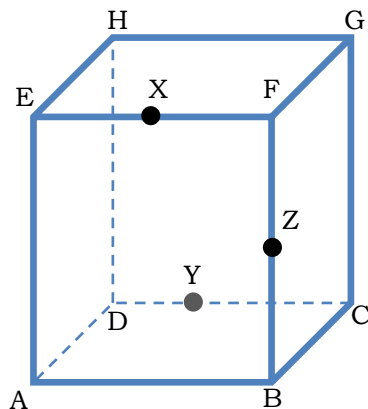


Figure 41: Cube with 3 points

The task is to find a cross section on this cube. A plane of section is defined by these 3 given points (for example X, Y and Z on Figure 41). Nowadays Slovak students solve this task by pencil and ruler in their workbooks. Teachers teach them the constructing method, which

consist of extending segments (mostly edges of the cube), using parallel segments, localizing intersections, etc., which leads to the determination of the rest of the points belonging to the section. This method is described in more detail in Appendix C, section C.3 Cube Section Construction Method.

To allow the user to do the same task but on the screen of a computer, we need to provide tools for an interaction which consists of individual elementary steps of the construction process.

System description

The application allows the user **to translate, rotate and zoom** the scene, so he/she can look at it from all angles. As with the constructing method, the application offers:

1. Definition of a **new point** on an intersection (for teachers there is the possibility to add points anywhere on the cube – on its vertex, edge or face – as the first three points for the task)
2. Insertion of a **new segment** defined either by two points or as a parallel line (with some other segment) passing through an existing point
3. **Extension of** an existing **segment** in one of its two directions
4. **Coloring of** a plane (either one plane of the cube, an auxiliary plane or the section plane)

Additional features are:

- The system activates snapping object after clicking the mouse (to preserve the educational character of application) and not before as it is usual.
- The system gives a feedback on the correctness of the solution.
- The system does not allow the students to edit the scene at the same level as teachers can.

User description

According the curriculum for eight-year grammar schools (Slovak Ministry of Education, 1997), secondary vocational school (Reiterová & Grmanová, 2005) and for four-year specialized secondary schools (Černek, 2005) approved by the Slovak Ministry of Education, the mathematics is a compulsory course and the solid geometry (which contains also the cross section of a cube) is taught in the penultimate year of the study. This establishes the primary target group to 17-19 years old students (users).

The secondary target group is the age range from 17 up within high school and technical university students, since they come into contact with abstract 3D objects – either some type of construction or more complex 3D objects visualization.

There are also other types of users - the teacher and the system administrator, but they are not relevant for our case study, therefore no detailed description is included.

Environment description

Dealing with educational 3D editor, the environment can be either a class, where each student works with a computer or it can be a home PC.

7.2 Evaluation

Our third goal was to verify whether different methods of visualized information increases usability of 3D graphical editors. The main emphasis was on graphical hint for a hovered object, which previews the consequence of object's selection. To verify this we set following hypotheses:

1. *Some users will notice the keyboard shortcuts on the tool buttons and try to use them, but they will prefer use the “special” keys for most often used tools (especially those users, who are more familiar with 3D editors).*
2. *Users will appreciate the action preview during scene editing, which visualize the consequence of their action.*
3. *Users will like grouped buttons (into modes), where system automatically detects the user's intention.*
4. *Users will appreciate the representation of infinite object in finite form, where these will be accommodated (“clipped”) to the preferred size. The same goes for finite objects.*

To evaluate these hypotheses we used following methods: observation the users during their work, questionnaires and interviews.

7.2.1 Systems Descriptions

We created two prototypes. Stereo3D (see Figure 42) and InteractiveCube (see Figure 43). Their design was influenced by our experiences obtained from project StereoVR Portal (Kovářová & Gregor, 2009). Both prototypes' scenes contain a cube with three points defining

the crossing plane. They share a common basic functionality and features:

- add a new point (only if it is an intersection) or delete it,
- add a line (connecting two points or a parallel one anchored to a point),
- delete, prolong or shorten the line
- define a cross section,
- check the correctness of the cross section,
- mouse wheel: zooming in/out the scene.
- no button pressed on mouse: dragging a selected object (if there is any), highlight a hovered object,
- status bar
- shortcuts

but they differ in details, which helped us to easily evaluate the contribution of individual features:

Stereo3D

- force the system show the cross section (as a hint),
- turn on/off anaglyph projection,
- left mouse button: press button in menu (toolbar or context menu - Figure 44), select an object, and translate the scene,
- right mouse button: rotating the scene (drag-and-drop),
- autorotation of a scene when unsuccessfully adding a new point.

InteractiveCube

- use drag-and-drop to extend or shorten the line/segment,
- use drag-and-drop to create a parallel line,
- use drag-and-drop to delete an object,
- left mouse button: press button, select an object, drag an object, rotate the scene, extend a line - Figure 45,
- right mouse button: default system function.

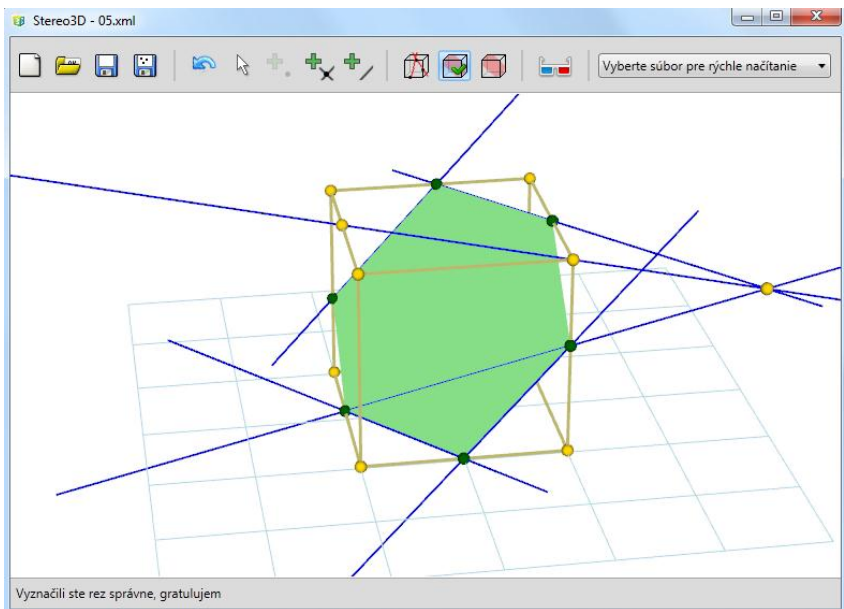


Figure 42: Screenshot of our pilot application Stereo3D

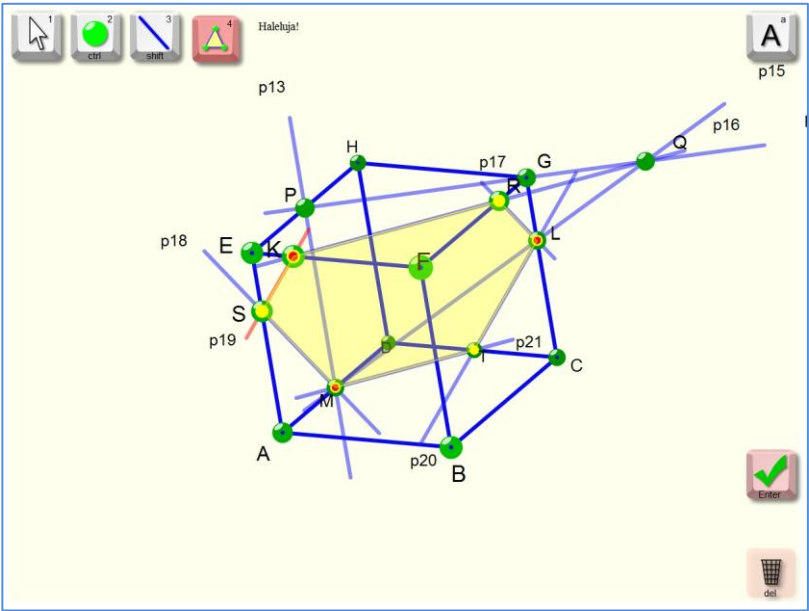


Figure 43: Screenshot of our pilot application InteractiveCube



Figure 44: Context menu for a line on Stereo3D

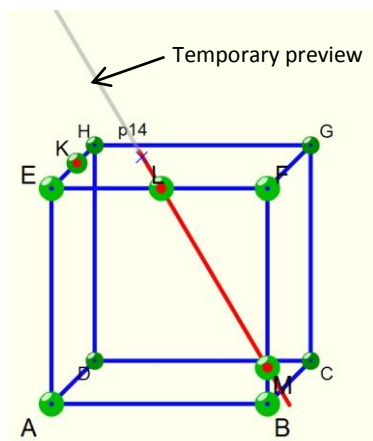


Figure 45: Preview of segment extension in InteractiveCube

7.2.2 Experiments

Our development included incremental prototyping – we pre-tested each part of Stereo3D interaction design to users from the secondary target group. We also consulted it with expert in didactics of mathematics – Mgr. Iveta Kohanová, PhD. who is Head of the Section of Didactics Mathematics at the Department of Algebra, Geometry and Didactics Mathematics, Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava, Slovakia and few other teachers from high schools. Finally we prepared a test for evaluation. We tested two different groups on Stereo3D prototype and the third on InteractiveCube prototype:

High school students – Stereo3D

Our first test group was the first year high school students from Grösslingová 18 in Bratislava (28 students). They did not pass yet the curriculum containing cube cross sections, which was also seen in some of the feedback questionnaire responses. We accommodated our testing to their level of knowledge, so their work with the application was not negatively influenced.

Before starting the test on high school, the purpose and objectives of our application testing was explained to the students. Since they had not studied that curriculum yet, we explained them elementary

principles in short, so they had the basic idea and could better understand what was expected in our application. We prepared a few simple tasks on cube section for them (e.g., the section plane was defined by adjacent cube corners). Students were instructed how they can load the tasks to the application, how they can check the correctness of their solutions and how to preview the correct solution if they have no idea how it should look like. They received no additional information. Students had limited time to try to solve the tasks. After it, they received and filled feedback questionnaire (for the result see next subsection)

Future teachers – Stereo3D

The second group was future teachers, students of bachelor degree at the Faculty of Mathematics, Physics and Informatics Comenius University in Bratislava (11 students). The method of testing them was similar. However, since they have sufficient knowledge on cube sections, they just received information on how to load a task to our application. The prepared tasks were more complicated than in previous group. Their difficulty was incrementally increasing ⁴¹. One of the aims of the experiment was to test a comprehensiveness of our UI – we let the students to work with application from the teacher's perspective, i.e., they were not provided with further information => they had to find out themselves how to control the application. Students had a few minutes to test applications and examples of tasks and subsequently received the same questionnaires as the high school students (for the result see next subsection).

Average users – InteractiveCube

The third group of testers belongs to the secondary target group of users (23 testers). Majority of our testers had either mathematical or computer science education and most of them spend on computer more than 16 hours per day. They were on average 30 years old. The method of testing was again similar. Testers did not receive any instruction and had to find out themselves how to control the application. They had three tasks of different difficulty (including the most complex one). They were not limited in time but majority of them did not spent with testing more than 5 minutes. After the testing they filled the questionnaires (for the result see next subsection). These differ from the previous ones only

⁴¹ The most complex task is when points on three non-neighboring edges of the cube.

in detail; the questions were accommodated to InteractiveCube prototype functionality.

Questionnaires

The feedback questionnaire was divided into four areas:

- *Look & Feel* – how do they like it
- *Arrangement* – a good arrangement of elements is important so the user recognizes the features of the program without the assistance and it has to be easy to navigate.
- *Controls* – they are the most important part from the user's point of view. If application has problematic controls or the user does not know or understand how to control it, it has lost every chance for further using.
- *Status Bar* – its aim was to provide on-going information and application feedback to the user. To make it more noticeable, the status bar flashed every time its information was changed. For this reason, one of the questions was if the users noticed this flashing (noticed new information) and whether this flashing does not disturb them in their work.

The majority of questions offer answers scaled to a scale from 1 to 5 (grade), where 1 usually means the fulfilled expectation. Some of the questions offer only the yes/no answer, which we mapped on 1 and 5. If there was a different answer pair, we mapped it on 2 and 4. For the rare question of type: "What would you change?" there was an empty field (no numerical expression for this type). The full text of the questionnaire can be found in Appendix C, section C.1 Feedback questionnaires C.1 Feedback questionnaire and graph of answers in section C.2 Graph.

Since our graph is not easy to read, we decided to express the test results in quickly understandable form – we represent each answer v_j by value calculated as weighted average of all responses in test group:

$$v_j = \frac{\sum_1^5 i \cdot a_{i,j}}{\sum_1^5 a_{i,j}}$$

where v_j is value of j^{th} answer for the whole test group, i goes through all possible grades that can have a single answer from a single tester and a_{ij} is the number of testers, which graded their j^{th} answer by the grade i . Since i goes from 1 to 5, the closer is the value v_j to 1, the result is better, closer to 5, the worse.

For easier handling, we have created for our three groups following abbreviations: High school students – HS, future teachers – FT, average users – AU.

7.2.3 Test results

Our prototypes were accepted positively. The users considered it simple, intuitive and user friendly (HS: 2.3, FT: 1.6, AU: 2.0). The future teachers showed a big interest in it. The ability to check the correctness of the solution and the ability to demonstrate the correct solution was evaluated very positively.

In following paragraphs are briefly described our results using our u_j values. For detailed overview see Appendix C, section C.1 Feedback questionnaires, where next to each question for each group of testers is given its u_j .

High school students – Stereo3D

The total number of high school students was 28. For most of them our application was comfortable (HS: 2.0), clear, with appropriately spaced buttons (HS: 1.7). Their meaning constituted a problem for a few students, but it can be attributed to the fact that they have heard about the cross section for the first time just before starting the application, so they were not familiar with the construction method. Students considered program controls very nice (HS: 2.0) and intuitive (HS: 2.2), and with a few exceptions no one had any problem with adding objects to the scene. Rotating and moving scenes were evaluated very positively (HS: 1.3), and except for two students, the assignment of these functions to the right and left mouse button suited them. Quite a lot of students had problems to carry out the intended operation in the application (HS: 2.7), which may in part be attributed to the fact that they were not able to learn the cube section in the short time they had available for testing. We believe that the longer test application would reduce the percentage. More than half of the students noticed the status bar during the work (HS: 2.5) and over a third noticed the help information (HS: 2.5). The vast majority was not distracted by it during their work (HS: 4.6).

Future teachers – Stereo3D

We had 11 university students – future teachers. They evaluated the application very positively (FT: 1.6). Almost all elements of the application worked nicely and clearly (FT: 1.5), the buttons were conveniently arranged (FT: 1.5) and there was no problem to understand it (FT: 1.1). Controls were intuitive and comfortable (FT: 1.3), the only exception was adding objects to the scene. Shifting and rotation of the scene was natural and fully complied with the use of the left and right buttons (FT: 1.0-1.5). We can negatively evaluate the fact that almost a third of respondents reported that they sometimes felt that they could

not perform the intended operation (FT: 3.8). The origin of this problem may be in an additional pop-up menu that appears when the user point at an object in the scene after a short interval. One solution might be the reduction of this interval, which, however, could cause too frequent menu popping up when moving the mouse in the scene. The status bar was noticed by three quarters of respondents (FT: 1.8) and the vast majority indicated that it was not distracting at all (FT: 1.6). Information was seen helpful only by about a third of respondents (FT: 3.4), which is possibly caused by their education and background.

Average users – InteractiveCube

The last test was attended by 23 average users. They found the application pleasant enough (AU: 2.0). They considered it very well designed (AU: 1.9). A half of testers would appreciate bigger letters but this is only matter of HTML code. The application has well-arranged buttons (AU: 1.7), which they consider understandable (AU: 1.9), for some of them were old-fashion. There was a suggestion keep all buttons together.

Majority of users agreed, that they can control the application intuitively (AU: 1.8) and the controls are comfortable enough (AU: 2.0). They sometimes missed a delete function even when it was there. This was caused by conditional availability – only in the mode of selection. Moreover, some objects cannot be deleted, because they are a part of the task. Due to this the intuitiveness of deletion felt down to AU: 2.6.

Adding a new line (AU: 1.5) or a point (AU: 2.0) was considered as without problems. In case, the objects got overlapped, majority of users rotate the cube, very few used zoom in. 87 % of testers found out how to create a parallel line (it has to be dragged from the master line), the rest did not – one of the tester claimed he did not need it. Only one third of all users considered manipulation with it as clearly intuitive (AU: 2.4) and a half of them without any problem (AU: 2.4). A half of all testers found it out by a chance; other half read it in information bar. 91% of all users turn the scene during their work, but not everybody was with this model of rotation satisfied (AU: 2.1). This was caused by different habits from different systems.

A half of the users had during their work with system feeling that they can't do what they want. This was due to the nature of the test itself. The users were in process of discovering the system abilities. These feelings often changed when they reached the end of the testing session. Our testers were not fully satisfied with the feedback (AU: 2.3), because they for example expected the system to let them know, if there

is an intersection or not. This was not implemented so that we preserve its educational value. There were of course some relevant objections, too.

Majority (2/3) of the users noticed and tried shortcuts right at the beginning, but only a half of them (1/3 of all users) kept using them and found it a good feature.

Everybody liked the preview of extended line (AU: 1.1).

A half of the users tried to extend the line even by dragging it, but 60% of all users simply preferred extension by clicking.

Everybody except 2 users noticed the information bar (AU: 1.3) and they read it to learn how to control the application (AU: 1.7). However, only a half of them considered it helpful enough (AU: 2.0).

A half of the users tried to turn off the objects' labels (AU: 2.8).

Summary

The test results shows that the most experienced users were FT, since they already worked with similar system (GeoGebra) and they also knew the construction method for cube cross section. On the other side, the less experienced with the least knowledge were HS – in this group the most difficulties occurred.

Now coming back to our hypotheses, according the questionnaires results the evaluation shows, that:

1. *1/3 of users immediately and the second 1/3 at the very beginning of the application use noticed the shortcuts. 39% of all users even tried to use it. The most often used key was Ctrl. This proves our first hypothesis.*
2. *91 % of all users agreed, that it was great to have a preview during scene editing, which visualized the consequence of their action. This proves our second hypothesis.*
3. *Since only one user from all three groups complained about unclear possibilities, when not every function is visible in menu, this can indirectly indicate, that the rest of the users liked grouped buttons (into modes), where system automatically detects the user's intention.*
4. *We did not clearly show that the users will appreciate the representation of infinite object in finite form, because the only group that did not like it was group of FT which tested Stereo3D prototype. Here the function drag-and-drop for segment extension was not implemented. We assume this interaction style would solve this kind of problem also for FT. None of the testers of InteractionCube complained about shortness of segments/lines.*

This proved that for 39 % of users were able to control our application by shortcuts, which made their work quicker. The most used key Ctrl indicate rapid adoption of its style of interaction (temporary mode change - similar to Shift key for capital letters).

Since 91 % of users rated our graphical preview of possible action very positively, this is the clear proof that this method works not only theoretically but also practically. This method reduces the interaction time required to perform an extension line action – the most often action in the system.

We assume that our last two hypothesis, which we did not confirmed ultimately, contribute to comfortable and faster work with the system (it looked so during the testing), but there are further testing needed to prove it clearly.

7.3 Discussion

To propose improvements or new interaction methods in the field of 3D graphics educational applications is not as easy as it may seem. We already tried gestures (Kovárová & Polák, 2011), we invented an original pointer working with the depth, we used the anaglyph rendering (Kovárová & Gregor, 2009), etc. All these methods improve or facilitate communication between a user and a computer from a theoretical point of view. But theory is not always consistent with practice. Our previous testing has shown that some of our ideas are inconvenient for users, or otherwise difficult, or sometimes we just found out that people are comfortable and do not like to let go their old habits.

In this chapter we described our last idea. The experiment showed that, unlike our previous ideas, this time our approach was very successful. We believe that this is the result of our focus on the information the users perceive visually and we got inspired by solutions working in other areas.

We consider the benefits of the graphical hint preview in systems, where expected the user will quickly learn to use them, but will use it only for a short time. An example of it is our prototype. We assume that in the long term use of such system these previews can become annoying. This can be managed by allowing to disable these previews. The disadvantage of this solution is just that the hint preview will show up when the mouse cursor is in a position, where can be done given action. This is on one side of a very sensible, but on the other hand, the user is not aware of this action possibility until the cursor will be moved there. Since in our experiment we have not noticed this at all, we

assume, this disadvantage can be mastered by choosing appropriate environment.

During the questionnaires evaluation, we noticed that different people prefer different styles of interaction. Some like to combine the mouse and the keyboard, others not. Some prefer the drag-and-drop style, others prefer to click. We cannot implement only one style that suits everyone. An interesting result of testing, however, was that the users who use the mouse along the keyboard (shortcut keys / switch keys) learned very quickly to use the temporary switch mapped on Ctrl-key. Such switch can be used in any application. There is only a question, how to let the users know about this feature, how to motivate them to try it and then use it regularly. A problem may occur only when the user would want to switch between more than two modes / options. But such situation occurs very rarely, usually with the experts.

Unlike the other similar applications (see section 4.3.3), in our prototype we have two above mentioned unique features, which accelerate and facilitate the users' interaction with 3D editor for solid geometry. Moreover, according to the math teachers, the most valuable feature is the automatic correctness evaluation ability, which they never experienced in previous applications.

An important part of the presented results will be published and presented at Interaction Collaborative Learning Conference in Piestany on September 22nd 2001.

8. Conclusions

The aim of this thesis was to contribute to the field of Human Computer Interaction. We propose three different interaction methods, which creates or increases the usability. Each of these methods solves a specific problem. We therefore used different methods for their verification. In following paragraphs are repeated our goals, how we fulfilled them, what are their benefits, where else would the method be applicable or how it could be improved.

Goal 1: To propose and verify a new method of computer aided interaction with remote objects within an environment (off the computer screen) using common and affordable equipment (including input and output devices).

We focused on a new and undiscovered application of human-computer interaction using a laser pointer and image processing. Our pilot project icPoint offers star and space object recognition using a laptop, a web camera, a laser pointer and a glass pane. In icPoint we implemented a new method for interaction with a distant object. Our acceptance test demonstrated its practical applicability. This shows that the proposed method of determining the user's line of sight and subsequently the determination of the object of interest works correctly with an acceptable deviation (3 degrees).

Although it is complicated to set up the system, once it is set, it is easy to use. To facilitate the control of our applications (since the user is in unconventional conditions) we proposed also a laser pointer mouse cursor control in combination with voice recognition. Moreover, the user can read or listen to information about sky objects stored on a local computer and get new information through our web service.

An important part of the presented results has been published at ICETA 2007 (see Appendix A section A.1 Publications) and icPoint was awarded with Quality Seal in Europrix Top Talent Award competition (see Appendix A, section A.2 Awards).



The method for pointing at distant objects can be used in any other situation, e.g., pointing at mountain peaks, building roofs (city panoramas), or archeological parks. Here, however, the glass would be positioned vertically and calculations for determining the object of interest would differ as well as the objects' data representation. Any of these implementations could be extended to an examination mode, where the user is asked to show the object and the system would determine whether it was the right direction. We created such a system for pointing at objects within indoor spaces (Kovárová, Meszáros, & Zelman, 2008).

Goal 2: To propose and verify a method, which on the basis of observing the user's actions, stores his/her choices and thus reduces the demands on the user's interaction when retrieving web information.

Our experiments have shown that the UI adaptation using user model can significantly increase the user's efficiency while working with the widget. This demonstrated that our hypothesis is correct and the proposed method fulfills our goal. The efficiency, as one of the basic parts of usability, was increased because of reducing the time necessary for interaction (entering inputs) and generally reducing the number of interaction steps. When compared with other methods, in the best case, the user using our widget does not have to interact with the widget at all.

Although users found it difficult to start to use our widget, it is a matter of the provider. But after overcoming the initial widget unfriendliness, the users considered the widget a very pleasant and useful application. They especially liked voice announcements.

Right now the widget works only for one specific information source, but principally final implementation is dependent only on the data provider. The method itself is independent and can be implemented with the same result for any domain dealing with line departures.

Our method is unique because, besides using a user model, it is accessible without a single click, while any other solution requires at least one click, often several interaction steps. User models as well as widgets are becoming very popular. They are already used in many different domains and in different types of applications. It is not usual to combine them and our combination is very specific.



An important part of the presented results has been published at international venues endorsed by ACM and IEEE in 2008 and 2010 (see Appendix A section A.1 Publications).

Presently we participate in the project iTransit, which provides the closest departures and even more functionality. iTransit already works with the user's context: time and position, which creates a comfortable interface. E.g., after one click the user can get the required information and that makes this solution comparable with ours. We plan to implement also a user model for repetitiveness and maybe also as a widget for different mobile operating systems.

Since any widget can be personalized, it can adjust itself to best serve the user, and thus making the retrieving of information more comfortable and quicker. In our case this adaptation is achieved by monitoring the user's choices and storing the number of selections for each choice in the user model. This method can be applied in any other application domain that deals with regular departures, e.g., logistics or catering. It can be also used in any application domain where repetitiveness is present, e.g., automatic functionality invocation if the application notices a sequence, which the user is repetitively calling.

Goal 3: Verification of increasing usability resulting from different methods of visualized information in 3D graphical editors. The main emphasis was on graphical hint for a hovered object, which previews the consequence of object's selection.

Our experiments showed that, our approach was very successful. 91 % of users rated our graphical hint preview of possible action very positively. This is the proof that this method works not only theoretically but also practically. This method reduces the interaction time required to perform an extension line action – the most often action in the system.

We consider the benefits of the graphical hint preview in systems, where expected the user will quickly learn to use them, but will use it only for a short time. An example of it is our prototype. We assume that in the long term use of such system these previews can become annoying. This can be managed by allowing to disable these previews. The disadvantage of this solution is just that the hint preview will show up when the mouse cursor is in a position, where can be done given action. This is on one side of a very sensible, but on the other hand, the user is not aware of this action possibility until the cursor will be moved there. Since in our experiment we have not noticed this at all, we

assume, this disadvantage can be mastered by choosing appropriate environment.

During the questionnaires evaluation, we noticed that different people prefer different styles of interaction. Some like to combine the mouse and the keyboard, others not. Some prefer the drag-and-drop style, others prefer to click. We cannot implement only one style that suits everyone. An interesting result of testing, however, was that the users who use the mouse along the keyboard (shortcut keys / switch keys) learned very quickly to use the temporary switch mapped on Ctrl-key. Such switch can be used in any application. There is only a question, how to let the users know about this feature, how to motivate them to try it and then use it regularly. A problem may occur only when the user would want to switch between more than two modes / options. But such situation occurs very rarely, usually with the experts.

Unlike the other similar applications (see section 4.3.3), in our prototype we have two above mentioned unique features, which accelerate and facilitate the users' interaction with 3D editor for solid geometry. Moreover, according to the math teachers, the most valuable feature is the automatic correctness evaluation ability, which they never experienced in previous applications.

An important part of the presented results will be published and presented at Interaction Collaborative Learning Conference in Piestany on September 22nd 2011.

References

- 3D Realms. (1996). *Duke Nukem 3D*. Retrieved 2011, from 3D Realms:
<http://www.3drealms.com/duke3d/>
- Agemsoft. (2011). *Naucteviac.sk*. Retrieved 2011, from Planéta vedomostí (The Planet of Knowledge): <http://www.naucteviac.sk/>
- Ahlborn, B. A., Thompson, D., Kreylos, O., Hamann, B., & Staadt, O. G. (2005). A practical system for laser pointer interaction on large displays. *Proceedings of the ACM symposium on Virtual reality software and technology (VRST '05)* (pp. 106-109). New York, NY, USA: ACM.
- AID-CREEM. (2009). *Geoplan-Geospace*. Retrieved 2011, from <http://www.aid-creem.org/geoplace.html>
- Ajzen, I. (1985). From intentions to actions: A theory of planned behavior. In J. Kuhl, & J. Beckmann, *Action control: From cognition to behavior* (pp. 11-39). Berlin: Springer-Verlag.
- Ajzen, I., & Fishbein, M. (1980). *Understanding Attitudes and Predicting Social Behavior*. Prentice Hall, Edgelywood Cliffs, NJ: Pearson.
- Apple. (2011). *Dashboard Widgets*. Retrieved 2011, from <http://www.apple.com/downloads/dashboard/>
- Autodesk, Inc. (2010). Autodesk Maya 2011 - Getting Started. Retrieved from http://images.autodesk.com/adsk/files/3ds_max_2011_tutorials.pdf
- Autodesk, Inc. (2010). Autodesk 3ds Max 2011. Retrieved from http://images.autodesk.com/adsk/files/3ds_max_2011_tutorials.pdf
- Bezerianos, A., & Balakrishnan, R. (2005). The vacuum: facilitating the manipulation of distant objects. *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 361-370). New York, NY, USA: ACM.
- Boström, F., Nurmi, P., Floréen, P., Liu, T., Oikarinen, T.-K., Vetek, A., et al. (2008). Capricorn - an intelligent user interface for mobile widgets. *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services* (pp. 327-330). Amsterdam, The Netherlands: ACM.

- Brewster, S. (1994). Providing a structured method for integrating non-speech audio into human-computer interfaces. *PhD thesis*. University of York, Human-Computer Interaction Group, Department of Computer Science.
- Brigham, E. O. (1988). *The fast Fourier transform and its applications*. Upper Saddle River, New Jersey: Prentice-Hall, Inc.
- Brusilovsky, P., & Millán, E. (2007). User models for adaptive hypermedia and adaptive educational systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Ed.), *The adaptive web. LNSC 4321*, pp. 3-53. Berlin, Heidelberg: Springer-Verlag.
- Caceres, M. (2008, Q1). *Widgets 1.0: The Widget Landscape*. Retrieved 03 05, 2011, from W3C : <http://www.w3.org/TR/widgets-land/>
- Caltrox Educational Software. (2011). *Geometry Master 3.0*. Retrieved 2011, from Caltrox Educational Software: <http://www.caltrox.com/products/geometry.htm>
- Carmichael, J. L. (2011). *Etched Glass Sundials 21st Century*. Retrieved 2011, from Stained glass sundials: http://advanceassociates.com/Sundials/Stained_Glass/sundials_EGP.html
- Cohen, C. A., & Hegarty, M. (2007). Sources of Difficulty in Imagining Cross Sections of 3D Objects. *The 29th Annual Conference of the Cognitive Science Society*. Nashville, Tennessee, USA.
- Constantine, L., & Lockwood, L. (1999). *Software for Use: A Practical Guide to the Essential Models and Methods of Usage-Centered Design*. Reading, MA: Addison-Wesley.
- Constantine, L. L., & Lockwood, L. A. (1996). Usage-centered software engineering: New models, methods, and metrics. *Proceedings of the 1996 International Conference on Software Engineering: Education and Practice (SEEP '96)* (pp. 2-9). Los Alamitos, CA: IEEE Computer Society.
- Crocodile Clips. (2000-2011). Retrieved 2011, from Yenka: <http://yenka.com/>
- Černek, P. (2005). *Učebné osnovy pre stredné odborné školy - štvorročné štúdium. Matematika. CD-2004-16970/33680-1:092 (The curriculum for secondary schools - four year study. Mathematics)*. Retrieved 2011, from Učebné osnovy predmetov pre skupinu študijných odborov SOŠ: http://www2.statpedu.sk/Pedagogicke_dokumenty/SOS/Osnovy/Matematika_SOS.doc
- Davis, F. D. (1989, September). Perceived Usefulness, Perceived Ease Of Use, And User Acceptance Of Information Technology. *MIS Quarterly*, 13(3), pp. 319-340.
- Davis, F. D., Bagozzi, R., & Warshaw, P. (1989). User acceptance of computer technology: A comparison of two theoretical models. *Management Science*, 35(8), pp. 982-1003.
- de Cotret, S., & de Cotret, P. R. (2005). *Cabri 3D User Manual*. Cabrillog.

- Dix, A., Finlay, J., Abword, G. D., & Beale, R. (2004). *Human-Computer Interaction (3rd ed.)*. Scotprint, Haddington, UK: Pearson Education Limited.
- Diz, O. V. (2010). *Blender 2.5 - Basic Interface*. Retrieved 2011, from www.blendnuts.com: <http://www.blendnuts.com/2010/06/blender-25-interface.html>
- Dumitrascu, S. (2000-2011). *Geometria - Interactive Geometry Software*. Retrieved 2011, from <http://www.geocentral.net/geometria/en/>
- Fishbein, M., & Ajzen, I. (1975). *Belief, Attitude, Intention and Behavior: An Introduction to Theory and Research*. Reading, MA: Addison-Wesley.
- Fishkin, K. P. (2004, September). A taxonomy for and analysis of tangible interfaces. *Personal and Ubiquitous Computing*, 8(5), 347-358.
- Gates, M. (2009). *Stellarium User Guide*. Retrieved 2011, from Stellarium: http://downloads.sourceforge.net/stellarium/stellarium_user_guide-0.10.2-1.pdf
- Goebel, A. (2008). *Documentation for Archimedes Geo3D*. Retrieved July 14, 2011, from raumgeometrie.de: <http://raumgeometrie.de/>
- Golden Oct. Computer Network Service Co.,Ltd. (2011). *Mathsay*. Retrieved 2011, from <http://www.mathsay.com/>
- Google. (2009). *Google Desktop*. Retrieved 2011, from <http://desktop.google.com/>
- Google. (2011). New to Google SketchUp - Video Tutorials. Retrieved 2011, from http://sketchup.google.com/intl/en/training/videos/new_to_gsu.html
- Hajduk, A., Hajduková, M., Chochol, D., Paľuš, P., Pittich, E., Porubčan, V., et al. (1987). *Encyclopedia of Astronomy (Encyklopédia astronómie)*. Bratislava: Obzor.
- Han, M., & Park, P. (2009). A study of interface design for widgets in web services through usability evaluation. *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human* (pp. 1013-1018). Seoul, Korea: ACM.
- Hewett, Baecker, Card, Carey, Gasen, Mantei, et al. (2009, July 29). *Definition and Overview of Human-Computer Interaction*. Retrieved February 14, 2011, from ACM SIGCHI Curricula for Human-Computer Interaction: <http://old.sigchi.org/cdg/cdg2.html>
- Hom, J. (1996-2003). Retrieved February 07, 2011, from The Usability Methods Toolbox: <http://usability.jameshom.com/> and <http://www.usabilityhome.com/>
- Chávez, F. d., Vega, F. F., Olague, G., & Montero, J. L. (2008). An independent and non-intrusive laser pointer environment control device system. *Proceedings of the 5th international conference on Pervasive services (ICPS '08)* (pp. 37-46). New York, NY, USA: ACM.

- Choe, E. K., Shinohara, K., Chilana, P. K., Dixon, M., & Wobbrock, J. O. (2009). Exploring the design of accessible goal crossing desktop widgets. *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems* (pp. 3733--3738). New York, NY, USA: ACM.
- Id Software. (1993). *Doom 3*. Retrieved 2011, from Id Software: <http://idsoftware.com/games/doom/doom3/index.html>
- Id Software LLC. (2002). *Wolfenstein 3D*. Retrieved 2011, from Id Software: <http://idsoftware.com/games/wolfenstein/wolf3d/index.html>
- Insam, E. (2002). *IrDA INTERFACING TO PCs*. Retrieved March 21, 2011, from Electronics World: <http://www.eix.co.uk/Articles/IrDA/Welcome.htm>
- International GeoGebra Institute. (2011). *Geogebra*. Retrieved 2011, from <http://www.geogebra.org/>
- Interplay. (1995). *Descent*. Retrieved 2011, from Interplay: <http://interplay.com/games/descent.php>
- Ishihara, M., & Ishihara, Y. (2006). An approach to remote direct pointing using gray-code. *Proceedings of the working conference on Advanced visual interfaces* (pp. 75-78). New York, NY, USA: ACM.
- Ivory, M. Y., & Hearst, M. A. (2001, December). The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys*, 33(4), 470-516.
- Jackiw, N. (2009). *The Geometer's Sketchpad v5.0*. Emeryville: Key Curriculum Press.
- Kaufmann, H. (2009). Dynamic Differential Geometry in Education. *Journal for Geometry and Graphics*, 13(4), 131-144.
- Kaufmann, H., & Schmalstieg, D. (2002). Mathematics and geometry education with collaborative augmented reality. *ACM SIGGRAPH 2002 conference abstracts and applications* (pp. 37-41). San Antonio, Texas: ACM.
- Kaufmann, H., Schmalstieg, D., & Wagner, M. (2000). Construct3D: A Virtual Reality Application for Mathematics and Geometry Education. *Education and Information Technologies*, 5(4), 263-276.
- Kieras, D. (2001). *Using the keystroke-level model to estimate execution*. Ann Arbor: Department of Psychology, University of Michigan.
- Kim, N., Lee, S., Lee, B., & Lee, J. (2007). Vision Based Laser Pointer Interaction for Flexible Screens. In J. Jacko (Ed.), *Human-Computer Interaction. Interaction Platforms and Techniques. Lecture Notes in Computer Science 4551*, pp. 845-853. Springer Berlin / Heidelberg.
- Kortenkamp, U. (1999). Foundations of Dynamic Geometry. *PhD Thesis*. Institut für Theoretische Informatik, ETH Zurich.
- Kortenkamp, U. H., & Richter-Gebert, J. (2000). *User Manual for the Interactive Geometry Software Cinderella*. Heidelberg: Springer.

- Kovárová, A., & Gregor, J. (2009). Solid Geometry and Virtual Reality. *SCO 2009, Proceedings of 6th conference on electronic support of education*, (pp. 93-98). Brno.
- Kovárová, A., & Polák, M. (2011). Virtual Reality Interaction via Mouse Gestures. *Spring Conference on Computer Graphics SCCG 2011 in cooperation with ACM and Eurographics* (pp. 43-46). Viničné: Comenius University.
- Kovárová, A., & Szalayová, L. (2010). Semantics in the Field of Widgets: A Case Study in Public Transportation Departure Notifications. In P. M. M. Wallace (Ed.), *Semantics in Adaptive and Personalised Services: Methods, Tools and Applications* (Vol. Studies in Computational Intelligence 279, pp. 93-107). Berlin Heidelberg: Springer.
- Kovárová, A., Meszáros, V., & Zelman, A. (2008). Laser Pointer Map Interaction. *Virtual University*. Bratislava: STU v Bratislave.
- Kules, B. (2000). User Modeling for Adaptive and Adaptable Software Systems. *ACM Conference on Universal Usability*. Arlington, VA, USA: ACM.
- Kurz, D., Hantsch, F., Grobe, M., Schiewe, A., & Bimber, O. (2007). Laser Pointer Tracking in Projector-Augmented Architectural Environments. *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality* (pp. 1-8). Washington, DC, USA: IEEE Computer Society.
- Leuf, B., & Cunningham, W. (2001). *The Wiki way: quick collaboration on the Web*. Boston, Massachusetts, USA: Addison-Wesley Longman Publishing Co., Inc.
- Lewis, C., Polson, P., Wharton, C., & Rieman, J. (1990). Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces. *Proceedings of CHI (Seattle, Washington, April 1 - 5, 1990)* (pp. 235-242). New York: ACM.
- Long, Jr., A. C., Landay, J. A., & Rowe, L. A. (1999). Implications for a gesture design tool. *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit* (pp. 40-47). Pittsburgh, PA, USA: ACM Press Addison-Wesley.
- Marinos, D., Geiger, C., Schwirten, T., & Göbel, S. (2010). Multitouch navigation in zoomable user interfaces for large diagrams. *ITS '10 ACM International Conference on Interactive Tabletops and Surfaces* (pp. 275-276). New York, NY, USA: ACM .
- Math Forum @ Drexel. (1994-2011). *Interactive Geometry Classroom Resources*. Retrieved 2011, from The Math Forum:
<http://mathforum.org/dynamic/classroom.html>
- Mäkelä, E., Viljanen, K., Alm, O., Tuominen, J., Valkeapää, O., Kauppinen, T., et al. (2007). Enabling the Semantic Web with Ready-to-Use Web Widgets. *International Semantic Web Conference*, (pp. 56-69).

- Microsoft. (2011). *Windows Desktop Gadgets*. Retrieved 2011, from MSDN: <http://msdn.microsoft.com/en-us/library/dd834142.aspx>
- Microsoft Corporation. (2011). *Web Slices*. (Microsoft Corporation) Retrieved 2011, from Internet Explorer 8 features: <http://windows.microsoft.com/en-US/internet-explorer/products/ie-8/features/easier?T1=ie8webslices>
- Moore, G. C., & Benbasat, I. (1991, September). Development of an Instrument to Measure the Perceptions of Adopting an Information Technology Innovation. *Information Systems Research*, 2(3), 192-222.
- Nielsen, J. (1993). *Usability Engineering*. San Diego, CA, USA: Academic Press.
- Nielsen, J. (2000, March 19). *Why You Only Need to Test with 5 Users*. Retrieved February 2, 2011, from Jakob Nielsen's Alertbox: <http://www.useit.com/alertbox/20000319.html>
- Nielsen, J., & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. *Proceedings of ACM INTERCHI'93 Conference on Human factors in computing systems* (pp. 206-213). Amsterdam, The Netherlands: IOS Press.
- Nielsen, J., & Mack, R. L. (1994). *Usability Inspection Method*. New York, NY: John Wiley & Sons.
- Nielsen, J., & Molich, R. (1990). Heuristic evaluation of user interfaces. *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people* (pp. 249-256). New York, NY, USA: ACM.
- Olsen, J. D., & Nielsen, T. (2001). Laser pointer interaction. *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 17-22). New York, NY, USA: ACM.
- Opera Software. (2010). *Creating your first Opera widget*. Retrieved 2011, from Dev.Opera: <http://dev.opera.com/articles/view/creating-your-first-opera-widget/>
- Parker, K. J., Mandryk, R. L., & Inkpen, K. M. (2005). TractorBeam: seamless integration of local and remote pointing for tabletop displays. *Proceedings of Graphics Interface 2005* (pp. 33-40). Victoria, British Columbia: Canadian Human-Computer Communications Society.
- Pixologic, Inc. (2011). Getting Started with ZBrush guide. Retrieved 2011, from http://download.pixologic01.com/download.php?f=Plugins/zbrush4-PDF/ZBrush4_Fundamentals.pdf.zip
- Pokorný, Z. (1998). *Astronomical Algorithms for Calculators (Astronomické algoritmy pro kalkulátory)*. Praha: Hvězdárna a Planetárium hl.města Prahy.
- Popa, A. (1999-2011). *GeomSpace User Manual*. Retrieved 2011, from GeomSpace: <http://sourceforge.net/projects/geospace/>
- Preece, J., & Keller, L. (1990). *Human-computer Interaction*. University Press, Cambridge, Great Britain: Prentice Hall.

- Raskin, J. (2000). *The humane interface: new directions for designing interactive systems*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.
- Reiterová, M., & Grmanová, A. (2005). *Učebné osnovy pre stredné odborné školy - štvorročné štúdium. Matematika. CD-2004-16970/33680-1:092 (Curricula for secondary vocational school - four year study. Mathematics)*. Retrieved 2011, from Učebné osnovy predmetov pre skupinu študijných odborov SOU - 4-ročné:
http://www2.statpedu.sk/buxus/docs/Pedagogicke_dokumenty/SOU-4/ucebne_osnovy/UO_SOU-4_matematika.pdf
- Rogers, E. M. (1995). *Diffusion of Innovations (4th ed.)*. New York: Free Press.
- Sauro, J. (2010, March 8). *Why you only need to test with five users*. Retrieved February 2, 2011, from Measuring Usability:
<http://www.measuringusability.com/five-users.php>
- Shizuki, B., Hisamatsu, T., Takahashi, S., & Tanaka, J. (2006). Laser pointer interaction techniques using peripheral areas of screens. *Proceedings of the working conference on Advanced visual interfaces (AVI '06)* (pp. 95-98). New York, NY, USA: ACM.
- Shneiderman, B. (1980). *Software psychology: Human factors in computer and information systems (Winthrop computer systems series)*. Winthrop Publishers.
- Shneiderman, B. (1998). *Designing the user Interface: Strategies for Effective Human-Computer Interaction (3rd ed.)*. USA: Addison Wesley Longman, Inc.
- Shubin, H. (1999). User models as a basis for Web design. *Proceedings of ACM/SIGCHI annual conference on Human Factors in Computing Systems. Workshop on Organizing Web Site Information*. Pittsburgh, PA, USA: ACM.
- Slovak Ministry of Education . (2011). *Na konferencii diskutovali učitelia z celého Slovenska o digitálnom učive Planéta vedomostí (Teachers from all over Slovakia discussed on conference about Planéta vedomostí curriculum)*. Retrieved 2011, from Slovak Ministry of Education - information for media: <http://www.minedu.sk/index.php?lang=sk&rootId=8537>
- Slovak Ministry of Education. (1997). *Učebné osnovy gymnázií - osemročné štúdium. Matematika. 1797/97-15 (Curriculum for eight-year high school. Mathematics)*. Retrieved 2011, from
http://www2.statpedu.sk/Pedagogicke_dokumenty/Gymnazia/8roc/Os_novy/osnovy_mat_gym_8r.doc
- Strasbourg astronomical Data Center. (2007). *Public Astronomical Catalogues and Lists*. Retrieved from <ftp://cdsarc.u-strasbg.fr/pub/cats/>
- Tuchinda, R., Szekely, P., & Knoblock, C. A. (2008). Building Mashups by example. *Proceedings of the 13th international conference on Intelligent user interfaces* (pp. 139-148). New York, NY, USA: ACM.

- van Dam, A. (1997, February). Post-WIMP user interfaces. *Commun. ACM*, 40(2), 63-67.
- Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. *MIS Quarterly*, 27(3), 425-478.
- Wang, J., Ding, Z., & Jiang, C. (2005). An Ontology-based Public Transport Query System. *First International Conference on Semantics, Knowledge and Grid, 2005* (pp. 62-64). Beijing: IEEE.
- Wong, J., & Hong, J. I. (2007). Making mashups with marmite: towards end-user programming for the web. *Proceedings of the SIGCHI conference on Human factors in computing systems. CHI '07*, pp. 1435-1444. New York, NY, USA: ACM.
- Yahoo! (2011). *Delicious*. Retrieved 2011, from The tastiest bookmarks on the web: <http://www.delicious.com/>
- Yahoo! (2011). *Konfabulator Reference Manual*. Retrieved 2011, from Yahoo! Widgets: <http://manual.widgets.yahoo.com/>

Appendix A

A.1 Publications

Publications with international recognition (B)

Kovárová Alena - Szalayová Lucia: Semantics in the field of widgets: a case study in public transportation departure notification. In: Semantics in Adaptive and Personalised Services: Methods, Tools and Applications. M. Wallace, Ph. Mylonas, I. Anagnostopoulos, M. Bielikova (Eds). Studies in Computational Intelligence Series, 10.3.2010, Springer, 2010. - ISBN 978-3-642-11683-4. p. 93-107.

Szalayová Lucia - Kovárová Alena: Personalized Widget for Public Transportation Departures. In: Semantic Media Adaptation and Personalization: Proceedings, Third International Workshop. Prague, Czech Republic, 15.-16.12.2008, IEEE Computer Society, 2008. - ISBN 978-0-7695-3444-2. p. 40-44.

Publications with national recognition (C)

Kovárová Alena – Michal Sokolský: Using Virtual Reality for Teaching Solid Geometry: A Case Study for a Cube Section. Accepted as oral presentation for International Conference on Interactive Collaborative Learning 2011, Piešťany, Slovakia

Kovárová Alena – Polák Marek: Virtual Reality Interaction via Mouse Gestures. In: Spring Conference on Computer Graphics SCCG 2011 in cooperation with ACM and Eurographics, Galbov Mlyn, Viničné April 28-30, 2011: Conference Materials and Posters. Bratislava: Comenius University, 2011. ISSN 1335-5694. p. 43-46.

Kovárová Alena - Mészáros Viktor - Zelman Andrej: Laser Pointer Map Interaction. In: Virtual University 2008: 9th International Conference. Bratislava, Slovak Republic, 11.-12.12.2008, STU v Bratislave, 2008. - ISBN 978-80-89316-10-6

Kovárová Alena - Czanner Silvester: Interactive simulations of Elementary Physical Experiments. In: Proceedings First Central European International Multimedia and Virtual Reality Conference. Veszprém, Hungary, 6.-8.5.2004, Veszprém University Press, 2004. - ISBN 963 9495 46 8. p. 109-116.

Kovárová Alena - Gregor Ján: Stereometria a virtuálna realita. In: SCO 2009
sborník 6. ročníku konferencie o elektronické podpore výuky. Brno,
Czech Republic, 2009, Masarykova univerzita, 2009. - ISBN 978-80-
210-4878-2. p. 93-98.

Škovran Ivan - Kovárová Alena: Tvorba a porovnanie náučnej 2D a 3D simulácie.
In: SCO 2007. Sharable Content Objects. Brno, Czech Republic, 30.-
31.5.2007, Masarykova univerzita, 2007.p. 113-118.

Other publications

Kovárová Alena: Digital Media Design: Survey of Requirements on
Interdisciplinary Study Program. In: Student Research Conference 2007.
3rd Student Research Conference in Informatics and Information
Technologies Bratislava, April 18, 2007, Bratislava : STU v Bratislave
FIIT, 2007. - ISBN 978-80-227-2631-3. p. 293-300.

Kovárová, Alena - Dobiš, Michal - Hlaváček, Vladimír - Ling Xuan - Jajcaj,
Michal - Lamoš, Dušan: icPoint - Interactive Night Sky Observation. In:
ICETA 2007 : 5th Int. Conference on Emerging e-Learning Technologies
and Applications. Stará Lesná, Slovak Republic, 6.-8.9.2007, Košice:
Elfa, 2007. - ISBN 978-80-8086-061-5. p. 255-260.

Kovárová, Alena - Šperka, Martin: Interactivity in Educational Applications
Using Virtual Reality. In: Student Research Conference 2006:
Proceedings in Informatics and Information Technologies, Vydavateľstvo
STU v Bratislave, 2006. - ISBN 80-227-2395-9. p. 293-300.

Kovárová, Alena: Multimedia Support for Teaching Physics. In: Virtual
University. VU'04. 5th International Conference: Proceedings. Bratislava,
Slovak Republic, Dec. 16-17, 2004, Bratislava: STU v Bratislave, 2004. -
ISBN 80-227-2171-9. p. 194-198.

Šperka, Martin - Kovárová, Alena: Digital Media Engineering and Design:
Proposals for the Interdisciplinary Study Program. In: ICETA 2007: 5th
Int. Conference on Emerging e-Learning Technologies and Applications.
Stará Lesná, Slovak Republic, 6.-8.9.2007, Košice: Elfa, 2007. - ISBN
978-80-8086-061-5. p. 341-346.

Šperka, Martin - Kovárová, Alena: Interdisciplinary and International Study
Programs in Digital Media Design and Engineering. In: Virtual University
VU '07: 8th International Conference. Bratislava, Slovak Republic,
Dec.13.-14. 2007, Bratislava: STU v Bratislave, 2007. - ISBN 978-80-
89316-09-0. p. 160-164.

Kovárová Alena: Dajme študentom virtuálnu skúsenosť. In proceedings: Trendy
v e-learningu. Prague, Czech Republic, Feb. 21.-22. 2005, 2005. - ISBN
80-01-03203-5. p. 39.

Šperka Martin - Drahoš Peter - Kovárová Alena: Computer Graphics and Related
Subjects at the FIIT STU and FI BVŠP. In: Future of Computer Graphics
Education - FCGE'09: Spring Conference on Computer Graphics.

Budmerice, Slovak Republic, 22.4.2009, 2009. ISBN 978-80-89313-46-4. p. 5-7.

A.2 Awards

6-10th place in the Czecho-Slovak ACM.SRC 2004 organized by the Czech chamber of ACM for the paper “Interactive simulations of Elementary Physical Experiments”
<http://acm.vsb.cz/student2004/index.html>



Official Top Talent Quality Seal in the EUROPRIX Top Talent Award 2007 for the project iPoint, coauthored as a part of a six member team



Tatra Banka Foundation award for leading an exceptional diploma thesis in IT field written by Lenka Litvová: “Multimedia application for mobile devices”



A.3 Research Projects

E-Talent Tatra Banka Foundation, 01/2011 - 12/2011, Development of applications for mobile devices, Assoc. prof. Michal Čerňanský (project iTransit)

KG 244-022STU-4/2010 (KEGA), Supporting the teaching process for parallel and distributed data processing, 01/2010 – 12/2011, Assoc. prof. Michal Čerňanský

E-Talent Tatra Banka Foundation, 01/2010 - 12/2010 The use of massive parallelism for mobile transport timetables, Assoc. prof. Michal Čerňanský (project iTransit)

VG 1/0848/08 (VEGA), Connectionist computational models for computer grid environment, 01/2008 – 12/2010, Assoc. prof. Michal Čerňanský

29079-IC-1-2005-1-DK-ERASMUS-PROG/3, Joint Degree in Media Development Engineering 9/2006 – 1/2010, Assoc. prof. Martin Šperka

VG 1/3103/06, Information infrastructure for the processing of knowledge scattered in a distributed environment, prof. 1/2006 – 12/2008, prof. Vladimír Vojtek

KEGA 3/3206/05, Interdisciplinary Study Program in Interactive Digital Media Design, 7/2005 - 12/2008, Assoc. prof. Martin Šperka

VG 1/0161/03 Information processing in a distributed environment of intelligent agents, 1/2003 - 12/2005, prof. Vladimír Vojtek

A.4 Supervised Theses and Projects

Bachelor Theses

Bc. Peter Tutko: Processing of physical experiments as educational multimedia, june 2007

Ing. Michal Smíšek: Multimedia Education via Internet, may 2008

Ing. Juraj Kollár: Development of web pages using XML and transformation languages, may 2008

Ing. et Ing. Rastislav Kršák: Development of web pages using XML and transformation languages, may 2008

Ing. Matúš Zjara: X3D and its capabilities in creating three-dimensional avatars, june 2009

Bc. Ondrej Ivančík: X3D and its capabilities of creation three-dimensional interactive scenes, june 2009

Ing. Andrej Kozák: X3D and its capabilities in modeling three-dimensional interactive scenes, june 2009

Bc. Juraj Jakabovič: Recognition of Music Score and its Consecutive Translation into MIDI, may 2010

Bc. Marek Takáč: Optical Recognition of Music Scores and Following Transfer into Midi File Format, may 2010

Diploma Theses

Ing. Ivan Škovran: Creation and comparison of educative 2D and 3D simulation, december 2006
presented on SCO2007

Ing. Lenka Litvová: Multimedia application for mobile devices, may 2009,
Dean's award sponsored by Tatra Banka Foundation, presented on IIT SRC 2009 – won Circuits and Systems, Communications Societies and Signal Processing Societies IEEE Chapter Prize

Ing. Martin Kozmon: Application for creating 3D models using sketch-based 2D modeling, may 2009
presented on IIT SRC 2009

Ing. Ján Gregor: Solid geometry and virtual reality, may 2009
presented on SCO 2009

Ing. Marek Polák: Forms of interaction in virtual reality usable in the process of education, may 2010

presented on SCCG 2011

Ing. Michal Drahoš: Interaction in Virtual Reality – Control, may 2010

Ing. Viktor Mészáros: Application for supporting common learning methods in classrooms with use of indirect control, may 2011

Ing. Samuel Števaňák: Interactive Learning Using Mobile Device, may 2011

Ing. Róbert Sopko: Personalized Interactive Education through Mobile Devices, may 2011

Ing. Peter Voroňák: Intelligent Image Resizing, may 2011

Ing. Michal Sokolský: Three Dimensional World of Solid Geometry, may 2011
presented on ICL2011

Team Projects

Virtual FIIT: Filip Hlaváček - Ján Hudec - Pavol Mešťaník - Matúš Novotný - Michal Palček - Rastislav Pečík - Ivan Polko, 2011,
presented on IIT-SRC 2011, TP Cup 2011 – 2nd place, Reaserchers'night 2011

Virtual FIIT: Lubomír Lackovič - Martin Mihalovič - Pavol Nágl - Marcela Polerecká - Martin Uhlík - Peter Voroňák, 2010,
presented on Reaserchers'night 2010

Interactive Weather Desktop Prototype: Čaučík - Tomáš Dankovčík - Peter jakubec - Martin Jakubéci - Ľuboš Ukrop - Martin Zachar, 2009,
presented on IIT-SRC 2009 – won The Czech ACM Chapter Prize, Reaserchers'night 2009

Bloodlezz: Sašo Kiselkov - Bianka Kováčová - Martin Kozmon - Lenka Litvová - Michal Poláčik - Jakub Tekeľ, 2008,
presented on IIT-SRC – won the Best Poster Award and Czechoslovak Section IEEE Award, Reaserchers'night 2008, Reaserchers'night 2009

icPoint: Michal Dobiš - Vladimír Hlaváček - Linh Hoang Xuan - Michal Jajcaj - Dušan Lamoš, 2007,
presented on IIT-SRC 2007, ICETA 2007, ACM SRC 2007 – won 4th place, Europrix TTA07 Award – won Quality Seal, Reaserchers'night 2008

Happy Guitar: Jozef Beňo - Martin Darula - Andrej Fiflík - Martin Darula - Martin Komara - Jozef Kriška - Ivan Škovran, 2006

Semestral projects

Adam Adámek: Multiplayer Game Portal, presented on IIT-SRC 2008

Lucia Szalayová: Public Transportation Departures Widget, presented on IIT-SRC 2008, SMAP 2008

Viktor Meszáros, Andrej Zelman: Laser Pointer Map Interaction, presented on IIT-SRC 2008, VU 2008, Reaserchers'night 2008, Reaserchers'night 2009

Michal Drahoš, Jozef Grexa: Voice Controlled Graphic Editor, presented on IIT-SRC 2008

Peter Mindek: Room information system, presented on IIT-SRC 2008

Peter Borga, Vladimír Mihál: User-friendly live score monitoring and match data visualization, presented on IIT-SRC 2008

Appendix B

B.1 Number of Test Users

In 2000 Nielsen published the widely cited web-article (Nielsen, 2000): "Why you only need to test with five users," summarizing the past decade's research and bringing a mathematical model which states that for the number of usability problems found in a usability test with n users the formula is:

$$N(1 - (1 - L)^n),$$

where N is the total number of usability problems in the design and L is the proportion of usability problems discovered while testing a single user. The typical value of L is 31%, averaged across a large number of projects they studied. Plotting the curve for $L=31\%$ gives the following result:

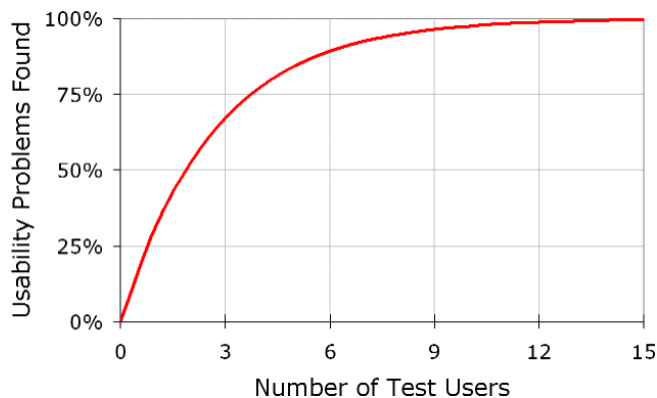


Figure 46: Graph showing dependency between number of test users and found usability problems (Nielsen & Landauer, 1993)

This graph (see Figure 46) was already known as the "parabola of optimism" (Nielsen & Landauer, 1993).

In 2010 Sauro wrote an article (Sauro, 2010) pointing out the problems of misinterpretation in Nielsen's model. His article reiterates

the important caveats made for the past three decades about the “magic” number 5:

1. You will definitely not see 85% of ALL problems; you will see only 85% of the more obvious problems (the ones that affect 31% or more of users).
2. The sample size formula only applies when you test users from the same population performing the same tasks on the same applications.
3. As a strategy, don't try and guess the average problem frequency. Instead, choose a minimum problem frequency you want to detect (p) and the binomial will tell you how many users you need to observe to have a good chance of detecting problems with at least that probability of occurrence.

If you approach sample sizes this way you avoid the problem of the variability in problem frequency and don't have to make any assumptions about the total number of problems in an interface.

B.2 icPoint

Minimal software requirements

- processor 1.6 GHz
- operation memory 512 MB
- graphical card supporting DirectX 9.0
- network card (needed for IP camera)
- free USB port (needed for Web camera)
- software: Windows XP OS, Net framework 2.0, Net framework 3.0, System libraries avicap32.dll, gdi32.dll, user32.dll, Microsoft SAPI 5.1 and Microsoft Windows Media Player 11

Voice commands

Finally, we decided to use Microsoft's SAPI engine⁴² to allow voice control and also for the speech synthesis. Using this SAPI we left the user with the choice of any English words to control application features.

The list of available voice commands for basic icPoint screen is in following table (Table 8); bold is used for the most often called commands. The rest of the voice commands – for the left side panel containing information about a sky object (see Figure 29) – are listed in Figure 9. To create higher intuitiveness of commands, we created for

⁴² <http://msdn.microsoft.com/en-us/library/ee125663%28VS.85%29.aspx>

some functions several possibilities. They were collected during different phases of our testing, where users were involved.

Table 8: Voice commands for basic icPoint screen

Command	Meaning
mouse down	Raises the mouse left button click event.
close application	Ends the application
quit application	
update background	This command is important for correct image recognition. Use it when the hardware is positioned properly and you are not pointing anywhere using the laser pointer (it is turned off). Use F5 key to do the same job.
update display	Redraws the celestial sphere in the application.
update sky	
use current sight	
show stars	
next star	Selects next (further from the center of the screen) sky object.
further star	
previous star	Selects the preceding (closer to the center of the screen) object.
closer star	
show object information	Displays the window containing information about the selected sky objects. (star or planet)
show star information	
show constellation information	Displays the window containing information about the constellation that contains currently selected star hide information Closes the window with the information about the sky objects.
increase magnitude	Increases the maximum visible magnitude by 0.5
decrease magnitude	Decreases the maximum visible magnitude by 0.5
increase view angle	Increases the display radius by 5 degrees
decrease view angle	Decreases the display radius by 5 degrees
start sky recognition	Executes the recognition of direction user's sight to stars
stop sky recognition	Stops the recognition of direction user's sight to stars
show constellation lines	Displays the constellation lines in the image of night sky.
hide constellation lines	Hides constellation lines in the image of night sky
show star names	Displays the names of the stars in the image of night sky
hide star names	Hides the names of the stars in the image of night sky
show constellation names	Displays the constellation name
hide constellation names	Hides the names of constellations

Table 9: Voice commands for icPoint left side panel, which contains information about a sky object

Command	Meaning
start mouse recognition	Executes the mouse cursor control
stop mouse recognition	Stops the mouse cursor
select item one	Selects the object 1 – 9 in the window with information
...	
select item nine	

about sky objects	Use this command to change selected expandable control (expander) at the same level
select item next	Selects following object in the window with the information about the sky object
select item previous	Selects preceding object in the window with the information about the sky object
select item lower	Selects first object contained within current expander in the window with the information about the sky object
select item upper	Selects first object contained within current expander in the window with the information about the sky object
expand item	Expands (shows its content) current object in the window with the information about the sky object
collapse item	Collapses (hides its content) current object in the window with the information about the sky object
use item link	Uses the link in current object – moves from star to the constellation and vice versa (changes the content of information window)
read item content	Reads the text in current object in the information window

B.3 Widget

Data model

The most important is to store lines, their stops and departures for terminal stops. While there is a difference in timetables depending on the day type, we enlarged our database with two small separated tables – public and school holidays (Figure 47).

The line table contains data about the line previously loaded by the system. There is a learning ability applied by lines - so one of the attributes is used to specify the incremental value of the line selection count.

The line stops table is loaded by data parsing of the left part of the schedule list. It contains information about stops of a respective line and time lag between each two upcoming stops in a route. Here the learning capacity of the system is done by incrementing the station selection count - selection of the station for specific line and direction.

The departure table in the database represents the departure times out of the base station - so the time of arrivals for a specific station is calculated using the initial departure time and summary of time lags until reaching the desired station. As departures are differentiated based on the actual day (working day, weekend, public holiday or school holiday) this feature is taken into consideration.

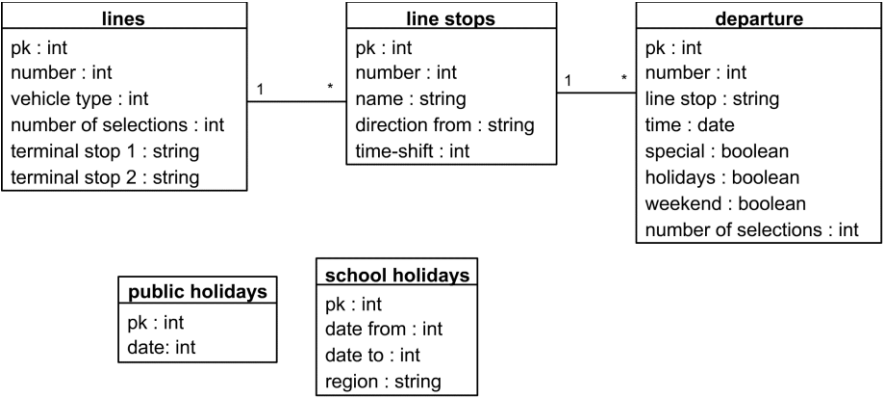


Figure 47: Physical data model of the widget database

The previously mentioned day differentiation is being done by recognizing a week day (working or not), whereas a special feature for recognition of public or school holidays is represented within separate tables with these special days. A list containing the school holidays is updated yearly - this list can be taken from the site of The Ministry of Education of the Slovak Republic⁴³. The attribute for region is necessary, because some school holidays in our country are region dependent.

⁴³ The web site of The Ministry of Education of the Slovak Republic, <http://www.minedu.sk>

Appendix C

C.1 Feedback questionnaires

Stereo3D application

General questions	FT v_j^{44}	HS v_j^{44}
Sex: male female		
How much time do you spend on a computer? 1-6 hours 7-15 hours 16 and more hours		
The appearance of the application		
1. Do you find the application pleasant? Very pleasant 1 ... 2 ... 3 ... 4 ... 5 doesn't look pleasant	1.6	2.3
2. Would you change its appearance? If yes, how?		
Design		
3. Do you find the application well designed? Very well designed 1 ... 2 ... 3 ... 4 ... 5 badly designed	1.5	2.0
4. Do you think the buttons are arranged appropriately? very appropriately 1 ... 2 ... 3 ... 4 ... 5 inappropriately	1.5	1.7
5. Did you understand the function of all buttons? yes 1 ... 2 ... 3 ... 4 ... 5 I had problem understanding them	1.1	2.2
6. Would you like to change something? If yes, please specify.		
Controls		
7. Can you control the application intuitively? intuitively 1 ... 2 ... 3 ... 4 ... 5 non intuitively	1.3	2.2
8. Are the application controls comfortable? comfortable 1 ... 2 ... 3 ... 4 ... 5 uncomfortable	1.8	2.0
9. Adding an intersection was without problems 1 ... 2 ... 3 ... 4 ... 5 problematic	2.2	2.8
10. Adding a point for the cross section was without problems 1 ... 2 ... 3 ... 4 ... 5 problematic	1.4	1.8
11. Adding a line was without problems 1 ... 2 ... 3 ... 4 ... 5 problematic	1.3	2.0
12. Adding a parallel line was without problems 1 ... 2 ... 3 ... 4 ... 5 problematic	2.0	2.2
13. Did you shift the scene during your work with the application? Yes no	1.4	1.1
14. Did you find the scene shifting natural?		

⁴⁴ In this column is given a weighted average of responses or their ratio.

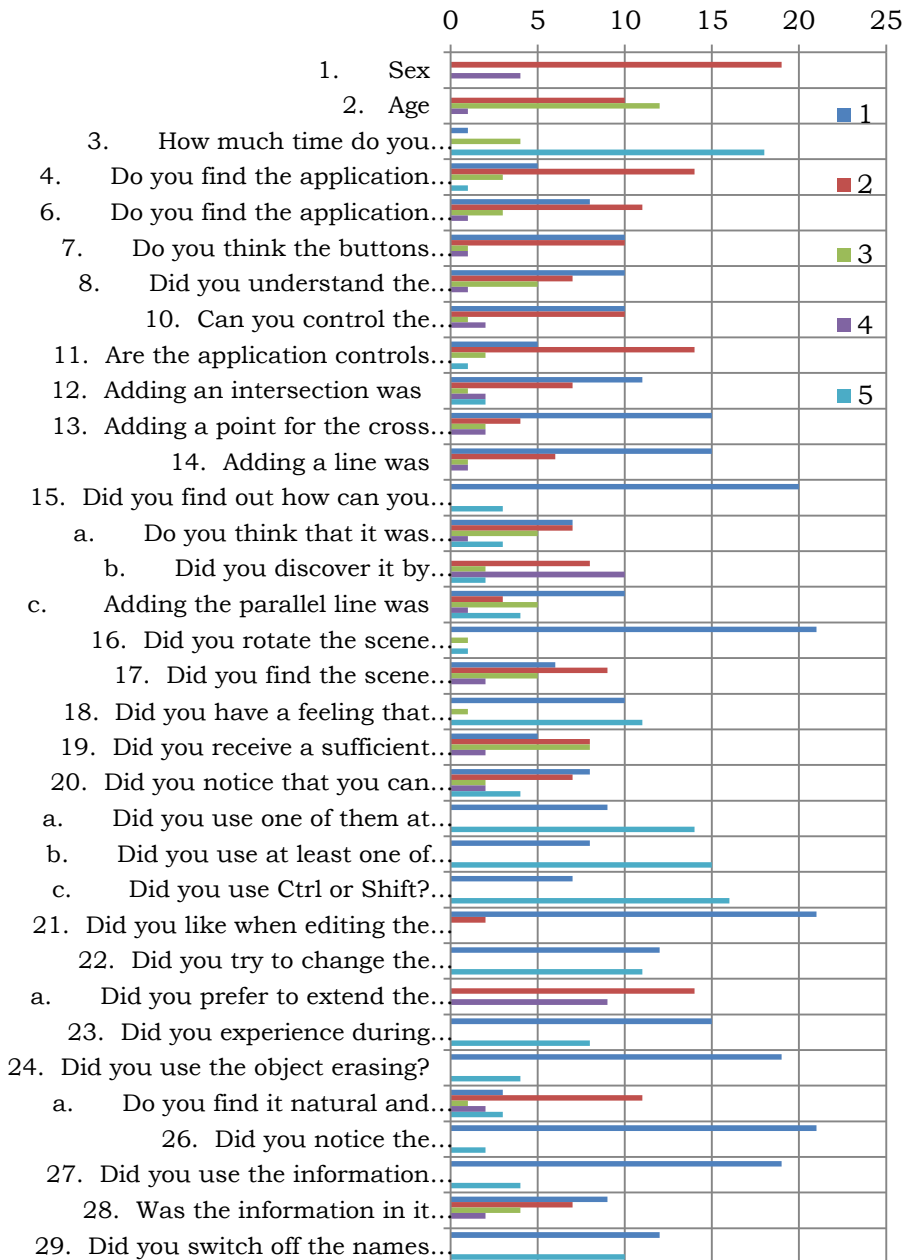
natural 1 ... 2 ... 3 ... 4 ... 5 unnatural	1.5	1.3
15. Did you find moving the scene with the left mouse button comfortable? Yes no	1.4	1.3
16. Did you rotate the scene during your work with the application? yes no	1.0	1.3
17. Did you find out the scene rotation natural? natural 1 ... 2 ... 3 ... 4 ... 5 unnatural	1.5	1.2
18. Did you find rotating the scene with the right mouse button comfortable? Yes no	1.0	1.3
19. Did you have a feeling that you can't do what you want because you didn't know how to perform it within the application? Yes no	3.8	2.7
20. Would you like to change something? If yes, please specify.		
Status bar		
21. Did you notice the status bar during your work? yes no	1.8	2.5
22. Did it distract you? Yes, I was distracted 1 ... 2 ... 3 ... 4 ... 5 No, it didn't distract me at all	1.6	1.4
22. Did the information from the bar help you during your work with the application? Yes, it was very helpful 1 ... 2 ... 3 ... 4 ... 5 No it didn't help me at all	3.4	2.5
23. Would you like to change something? If yes, please specify.		

InteractiveCube

General questions	v_j^{44}
1. Sex: male female	19:4
2. Age:	30
3. How much time do you spend on a computer? 1-6 hours 7-15 hours 16 and more hours	>16
Feel&Look	
4. Do you find the application pleasant? Very pleasant 1 ... 2 ... 3 ... 4 ... 5 doesn't look pleasant	2.0
5. Would you change its appearance? How?	
Design	
6. Do you find the application well designed? Very well designed 1 ... 2 ... 3 ... 4 ... 5 badly designed	1.9
7. Do you think the buttons are arranged appropriately? very appropriately 1 ... 2 ... 3 ... 4 ... 5 inappropriately	1.7
8. Did you understand the function of all buttons? yes 1 ... 2 ... 3 ... 4 ... 5 I had problem understanding them	1.9
9. Would you like to change something? If yes, please specify.	
Controls	
10. Can you control the application intuitively? intuitively 1 ... 2 ... 3 ... 4 ... 5 non intuitively	1.8
11. Are the application controls comfortable? comfortable 1 ... 2 ... 3 ... 4 ... 5 uncomfortable	2.0
12. Adding an intersection was without problems 1 ... 2 ... 3 ... 4 ... 5 problematic	2.0
13. Adding a point for the cross section was without problems 1 ... 2 ... 3 ... 4 ... 5 problematic	1.6
14. Adding a line was without problems 1 ... 2 ... 3 ... 4 ... 5 problematic	1.5
15. Did you find out how can you add a parallel line? yes no	1.5
a. Do you think that it was intuitive? yes 1 ... 2 ... 3 ... 4 ... 5 I had problems to find it out	2.4
b. Did you discover it by chance or did you read the text help? by chance reading the text help	10:12
c. Adding the parallel line was without problems 1 ... 2 ... 3 ... 4 ... 5 problematic	2.4
16. Did you rotate the scene during your work with the application? yes no	1.3
17. Did you find the scene rotation natural? natural 1 ... 2 ... 3 ... 4 ... 5 unnatural	2.1
18. Did you have a feeling that you can't do what you want because you didn't know how to perform it within the application? yes no	3.1
19. Did you receive a sufficient feedback from the application? Yes, I wasn't missing anything 1 ... 2 ... 3 ...4 5 I didn't have an idea what is the application doing.	2.3
20. Did you notice that you can use keyboard shortcuts?	

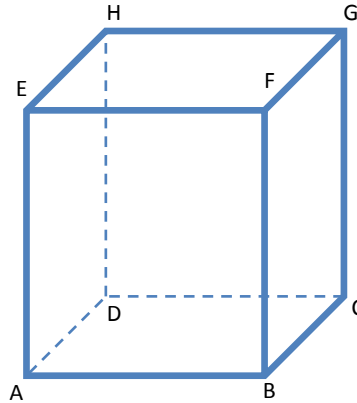
Yes, immediately 1 ... 2 ... 3 ... 4 ... 5 Are there any?	2.4
a. Did you use one of them at least once? Which one(s)? yes no these:	3.4
b. Did you use at least one of them repeatedly? Which one(s)? yes no these:	3.6
c. Did you use Ctrl or Shift? Have you found it user friendly? Yes no Yes no	3.8
21. Did you like (Was it easier for you to use the controls) that when editing the segments the application showed in grey color the planned extension of the relevant segment? Yes, that was great 1 ... 2 ... 3 ... 4 ... 5 I hated those pop ups.	1.1
22. Did you try to change the length of the segment using the drag and drop system? yes no	2.9
a. Did you prefer to extend the segment by clicking or by dragging it? Clicking dragging	14:9
23. Did you experience during the work that you were blocked by an object, which overlapped the object you wanted to work with? yes no	2.4
a. How did you solve this problem?	
24. Did you use the object erasing? yes no	1.7
a. Do you find it natural and comfortable? Natural 1 ... 2 ... 3 ... 4 ... 5 unnatural	2.6
25. Would you like to see a change in the controls? If yes, what?	
Information text	
26. Did you notice the information text during your work? yes no	1.3
27. Did you use the information from the text in order to find out how to control the application? yes no	1.7
28. Was the information in it helpful for your work with the application? helpful 1 ... 2 ... 3 ... 4 ... 5 not helpful	2.0
29. Did you switch off the names of the points and segments? yes no	2.8
30. Would you like to change something? If yes, please specify.	
Bug report:	
.....	

C.2 Graph



C.3 Cube Section Construction Method

This task is working with 6 basic planes, which define a cube:



1. ABCD – bottom plane - β
2. EFGH – up plane - υ
3. BCGF – right plane - ρ
4. ADHE – left plane - λ
5. ABFE – front plane - φ
6. DCGH – back plane - x

All of them together are called cube planes. There are parallel pairs: bottom and up, right and left, front and back.

The three set points define section plane. There are three rules, which are used to find the section:

1. If two points of the section belong to one of the cube's plane, connect them (you get line intersection of the cube plane and the section plane) and find intersections of this line and cube's edges (sometimes it is useful to find all four of them, where two belong to the cube's edges and two belong to extended cube's edges)

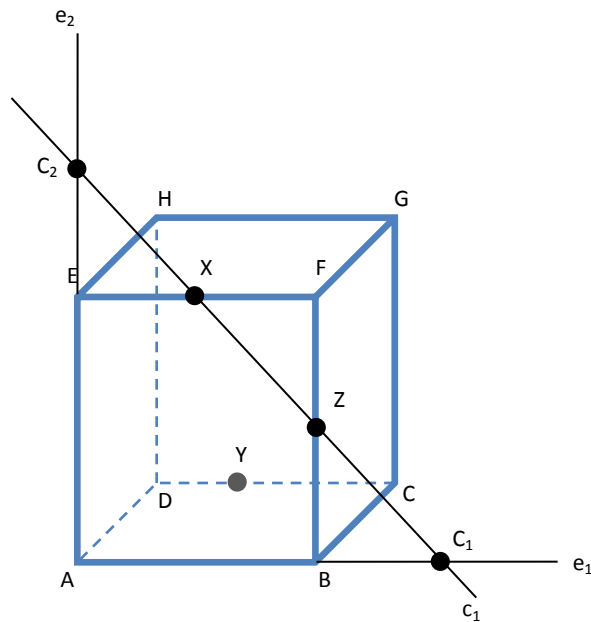


Figure 48: An example of the first construction rule

Construction steps:

$X, Z \in \varphi$

1. $c_1; c_1 = XZ \Rightarrow c_1 \in \varphi$ (section line)
2. $e_1; e_1 = AB \Rightarrow e_1 \in \varphi, \beta$ (extended edge)
3. $e_2; e_2 = AB \Rightarrow e_2 \in \varphi, \lambda$ (extended edge)
4. $C_1; C_1 = c_1 \cap e_1 \Rightarrow C_1 \in \varphi, \beta$ (section point)
5. $C_2; C_2 = c_1 \cap e_2 \Rightarrow C_2 \in \varphi, \lambda$ (section point)

2. If you already found the line intersection of the section plane with one of the cube's planes and you have one point of section plane lying on the parallel cube's plane, make a parallel line with this intersection line in this parallel cube's plane and find intersections of this line and the cube's edges

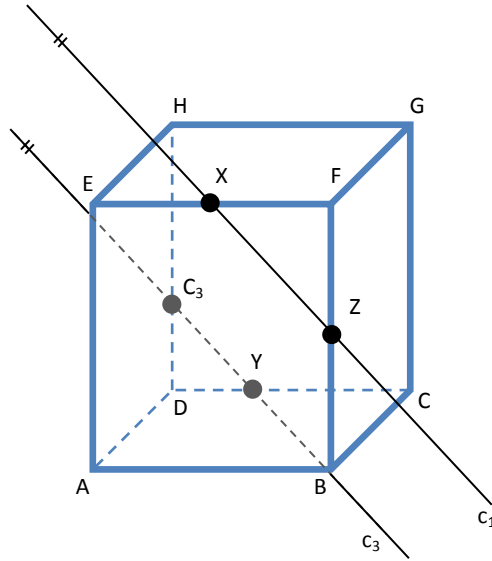


Figure 49: An example of the second construction rule

Construction steps:

$X, Z \in \varphi; \varphi \parallel x$

1. $c_1; c_1 = XZ \Rightarrow c_1 \in \varphi$ (section line)

2. $c_3; c_3 \parallel c_1 \Rightarrow c_3 \in x$ (section line)

3. $C_3; C_3 = c_3 \cap DH \Rightarrow C_3 \in x, \lambda$ (section point)

3. If there are no points belonging to one of the cube's planes, it is necessary to construct an auxiliary plane α , which is perpendicular to one of the cube's planes and two of set points belong to it (e.g. line c_4). Intersection of plane α and perpendicular cube's plane (e.g. β) is auxiliary line a , which is projection of line c_3 to α . Finally, the intersection of line c_4 and a is cutting point C_4 , which belongs to plane α as well as to plane β .

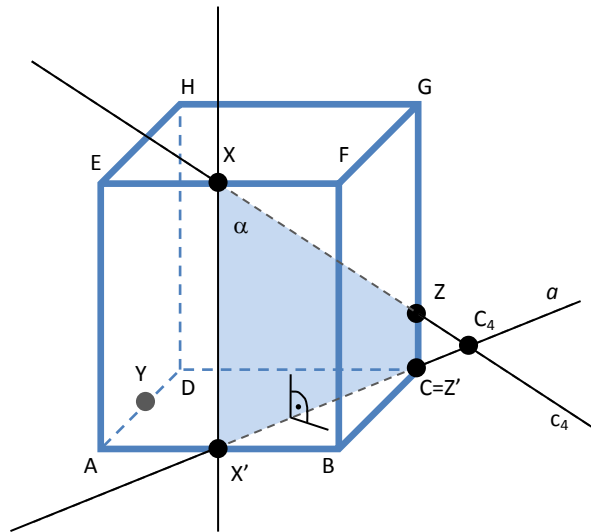


Figure 50: An example of the third construction rule

Construction steps:

$\alpha \perp \beta$; $X, Z \in \alpha$

1. X' ; $XX' \perp \beta$ (projection X to β)

2. Z' ; $ZZ' \perp \beta$ (projection Z to β , $Z' = C$)

3. α ; $\alpha = Z'ZXX'$ (auxiliary plane)

4. a ; $a = Z'X'$ (auxiliary line)

5. C_4 ; $C_4 = ZX \cap Z'X' \Rightarrow C_4 \in \alpha, \beta$ (section point)